Hierarchically Clustered Adaptive Quantization CMAC and Its Learning Convergence

S. D. Teddy, E. M.-K. Lai, *Senior Member, IEEE*, and C. Quek

***Abstract*—The cerebellar model articulation controller (CMAC) neural network (NN) is a well-established computational model of the human cerebellum. Nevertheless, there are two major drawbacks associated with the uniform quantization scheme of the CMAC network. They are the following: 1) a constant output resolution associated with the entire input space and 2) the generalization-accuracy dilemma. Moreover, the size of the CMAC network is an exponential function of the number of inputs. Depending on the characteristics of the training data, only a small percentage of the entire set of CMAC memory cells is utilized. Therefore, the efficient utilization of the CMAC memory is a crucial issue. One approach is to quantize the input space nonuniformly. For existing nonuniformly quantized CMAC systems, there is a tradeoff between memory efficiency and computational complexity. Inspired by the underlying organizational mechanism of the human brain, this paper presents a novel CMAC architecture named hierarchically clustered adaptive quantization CMAC (HCAQ-CMAC). HCAQ-CMAC employs hierarchical clustering for the nonuniform quantization of the input space to identify significant input segments and subsequently allocating more memory cells to these regions. The stability of the HCAQ-CMAC network is theoretically guaranteed by the proof of its learning convergence. The performance of the proposed network is subsequently benchmarked against the original CMAC network, as well as two other existing CMAC variants on two real-life applications, namely, automated control of car maneuver and modeling of the human blood glucose dynamics. The experimental results have demonstrated that the HCAQ-CMAC network offers an efficient memory allocation scheme and improves the generalization and accuracy of the network output to achieve better or comparable performances with smaller memory usages.**

***Index Terms*—Cerebellar model articulation controller (CMAC), hierarchical clustering, hierarchically clustered adaptive quantization CMAC (HCAQ-CMAC), learning convergence, nonuniform quantization.**

I. Introduction

THE human cerebellum is a brain region in which the neuronal connectivity is sufficiently regular to facilitate a substantially comprehensive understanding of its functional properties. It constitutes a part of the human brain that is important for motor control and a number of cognitive functions [1], including motor learning and memory. The human cerebellum is postulated to function as a movement calibrator [2], which is involved in the detection of movement error and the subsequent coordination of the appropriate skeletal responses to reduce the error [3]. The human cerebellum functions by performing associative mappings between the input sensory information and the cerebellar output required for the production of temporal-dependent precise behaviors [4]. The Marr–Albus–Ito model [3], [5] describes how the climbing fibers of the cerebellum assist this function by transmitting moment-to-moment changes in sensory information for movement control. Therefore, as a locomotive action progresses, the cerebellum is able to generate corrective signals to gradually reduce the movement error.

The cerebellar model articulation controller (CMAC) [6] is a neural network (NN) inspired by the neurophysiological properties of the human cerebellum and is recognized for its localized generalization and rapid algorithmic computations. As a computational model of the human cerebellum, CMAC manifests as an associative memory network [7], which employs error correction signals to drive the network learning and memory formation processes. This allows for advantages such as simple computation, fast training, local generalization, and ease of hardware implementation [2], and subsequently motivates the prevalent use of CMAC-based systems for system control and optimizations [8]–[12], modeling and control of robotic manipulators [13], [14], as well as various signal processing and pattern-recognition tasks [15]–[18]. The learning convergence of the CMAC network has also been established in [19]–[21].

As a functional model of the human cerebellum, CMAC operates based on the principle that similar inputs should produce similar outputs, while inputs that are dissimilar should invoke nearly independent outputs. As an associative memory, CMAC stores information locally and computes by employing a table-lookup operation, in which the contents are indexed by the inputs to the network. In the basic CMAC network, the network computing (memory) cells are equally divided to cover the whole input space. The CMAC inputs are then *quantized* to one of the discrete network cells to compute the indexes to retrieve the network output. We refer to this process as the uniform quantization of the input space.

A problem with the uniform quantization scheme arises when a higher output resolution is needed in certain segments of the target function. In these cases, the uniform quantization process will result in suboptimal memory space utilization and, in turn, the learned data will not be precisely modeled. Furthermore, due to the uniform quantization scheme, there is a tradeoff between the generalization capability and the modeling fidelity of the CMAC network. A small-sized CMAC is able to better generalize the characteristics of the training data but at a lower

Manuscript received May 9, 2006; revised January 15, 2007; accepted January 28, 2007.

S. D. Teddy and C. Quek are with the School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore.

E. M.-K. Lai is with the Institute of Information Sciences and Technology, Massey University, Wellington 6140, New Zealand (e-mail: e.lai@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2007.900810

output resolution, while a large-sized CMAC produces more accurate outputs at the expense of data generalization. These problems are in addition to the exponentially increasing CMAC memory size with the addition of each new input variable. It is, therefore, judicious to devise an efficient memory allocation scheme for the CMAC network to assign more memory cells to the input regions that require higher output resolution to enhance the CMAC memory utilization and to provide more accurate outputs with a reasonable degree of data generalization. The memory allocation process in a CMAC-based system refers to the construction of the CMAC computing structure (i.e., the quantization functions) to define the output resolution with respect to the different regions of the input space.

In the literature, there is currently a number of attempts to address the problems of uniform quantization in a CMAC-based system. Generally, these efforts can be classified into two main approaches. The first approach involves the use of multilayered CMACs of increasing resolutions [17], [22]–[25], while the second approach employs a CMAC network with varying quantization step-sizes [26]–[30]. These attempts are briefly discussed in Section III-C. However, these CMAC variants generally involve high computational complexity. They also lack a strong theoretical proof of the system's learning convergence, which is a desirable attribute for control and function approximation tasks.

In this paper, we propose the nonuniform quantization of the CMAC network based on hierarchical clustering. The resultant architecture is referred to as the hierarchically clustered adaptive quantization CMAC (HCAQ-CMAC) network. The objective of the HCAQ-CMAC network is to formulate a memory allocation procedure to enhance the storage efficiency, as well as to alleviate the generalization-accuracy dilemma of the CMAC network. The proposed network is inspired by the neurophysiology of the human brain, where the excess neurons and connections of the infant brain are gradually pruned and refined to form the precise wirings of the adult brain [31]. In addition, the learning process of the proposed HCAQ-CMAC network always converges when the learning rate is within a theoretical range.

The rest of this paper is organized as follows. Section II describes the neurophysiological aspects of the human cerebellum and the development process of the human brain which inspires the HCAQ-CMAC network. Section III outlines the basic principles of the CMAC NN, followed by a brief review of the existing CMAC nonuniform quantization schemes proposed in the literature. Section IV describes in details the proposed HCAQ-CMAC architecture. The proof of learning convergence of the HCAQ-CMAC network is established in Section V. Section VI evaluates the performance of the HCAQ-CMAC network against the basic CMAC and two other CMAC variants in two real-life applications, namely, automatic control of car maneuver and modeling the dynamics of the human glucose metabolic process. Section VII concludes this paper.

II. Human Cerebellum and Stages of Brain Development

The human cerebellum, or *little brain* in Latin, is a brain construct that is important for a number of motor and cognitive functions, including learning and memory [32], [33]. The most striking feature of the human cerebellum is the near-crystalline structure of its anatomical layout. However, despite its remarkably uniform anatomical structure, the cerebellum is divided into several distinct regions. Each of these regions receives sensory information from different parts of the brain as well as the spinal cord and projects to different motor systems. Such physical connectivity suggests that different regions of the human cerebellum perform similar computational operations but on different sensory inputs [4].

The cerebellum is provided with an extensive repertoire of information about the objectives (intentions), actions (motor commands), and outcomes (feedback signals) associated with a physical movement. There are two major sets of extra cerebellar afferents: the *mossy fibers* and the *climbing fibers*, both of which carry sensory inputs from the periphery as well as sets of motor commands-related information from the cerebral cortex [34]. The mossy fibers carry information originating from the spinal cord and the brainstem, while the climbing fibers originate from the inferior olivary in the medulla oblongata.

These cerebellar afferent inputs flow into the *granule cell* layer of the cerebellar cortex. The mossy fiber inputs, which carry both sensory afferent and cerebral efferent signals, are relayed by a massive number of granule cells. These granule cells work as expansion encoders by combining different mossy fiber inputs. Subsequently, each of the granule cells extends an ascending axon that rises up to the molecular layer of the cerebellar cortex as *parallel fiber*. These parallel fibers in turn serve as the inputs to the *Purkinje cells* at the cerebellar cortex. The Purkinje cells are the main computational units of the cerebellar cortex. The parallel fibers run perpendicularly to the flat fan-like dendritic arborization of the Purkinje cells, enabling the greatest possible number of parallel fibers and Purkinje cells contacts per unit volume. The Purkinje cells perform combinations of the synaptic inputs, and their axons carry the output of the cerebellar cortex downwards into the underlying white matter and subsequently to the *deep cerebellar nuclei*. The outputs of the deep cerebellar nuclei form the overall output of the cerebellum.

Memory formation in the cerebellum is facilitated by the information embedded in its synaptic connections. The cerebellum corresponds to an associative memory system that performs a nonlinear mapping from the mossy fiber inputs to the Purkinje cells' outputs. This mapping is functionally depicted as Fig. 1. The granule cell layer acts as an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells forms an array of modifiable synaptic weights of the cerebellar computing system. The Purkinje cell array subsequently forms the knowledge base of the cerebellum and generates the output of the cerebellar memory system by integrating the input synaptic connections.

Further, each of the diverse functions performed by the mature human nervous system, including the cerebellum, depends on the *precise* interconnections of its neurons that are formed through two stages of development, namely, the embryonic and the postnatal development stage. The embryonic development of the nervous system involves the generation of an overabundance of neurons and synaptic connections, followed by the pro-

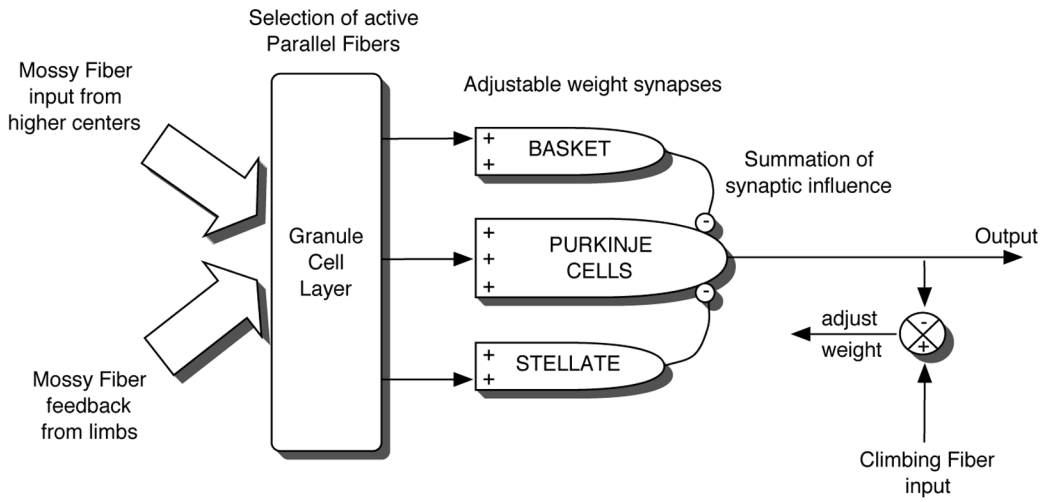


Fig. 1. Schematic diagram of the cerebellum (adapted from [2]).

grammed death of superfluous cells [35]. The synaptic elimination process involves *competition* among the neuronal cells [36], and as many as half of the initially generated neuronal connections are eliminated during the embryonic development stage. At the same time, the surviving neurons grow to be more complex and take over the functions of the eliminated ones.

This initial coarse pattern of neuronal connections is further refined during the postnatal development stage. In this stage, sensory information plays a critical role in strengthening and eliminating synaptic connections through a competitive process. Findings from neuroscience research [37], [38] have shown that learning as well as exposures to various stimuli greatly affect the patterns of the internal connections of the brain. This observation emphasizes the notion that the structural organizations and the neuronal mappings of the various brain regions are subjected to constant modifications based on adaptations (e.g., learning) and experiences. Experiences, however, function more than merely altering the synaptic connections. There are also evidences of an ongoing *synaptic reallocation* process [39], where neural cells are competitively shared by the brain's diverse functions.

With respect to the human cerebellum, there are evidences supporting the role of *experience-dependent plasticity* in cerebellar learning and memory formation. Essentially, two types of experience-dependent plasticity are observed in the cerebellum, namely, *synaptic plasticity* and *structural plasticity*. Synaptic plasticity refers to the modifiable synaptic strengths of the cerebellar circuitry that are achieved through the long-term depression (LTD) mechanism of the cerebellar learning process [40]–[42]. LTD alone, however, is not adequate for forming more permanent, long-term memories of procedural skills. Structural plasticity, on the other hand, refers to the alteration of the morphological structure of the neuronal interconnections in the cerebellum. Cerebellar structural plasticity studies have demonstrated that complex motor skill learning actually leads to an increase in the number of synapses within the cerebellar cortex [43]–[46]. It is primarily responsible for the formation of persistent and long-lasting memory traces in the human cerebellum. Currently, findings from physiological and biological brain studies have converged on the notion of experience being the key driving factor responsible for the specialization of the neuronal interconnectivity patterns in the human cerebellum.

III. CMAC as a Computational Model of the Human Cerebellum

The CMAC NN is a well-established computational model of the human cerebellum [6], [7]. The model was constructed to explain the information-processing characteristics of its biological counterpart. This section presents the basic computational principles of the CMAC NN and reviews some of the proposed nonuniform quantization techniques in the literature.

A. Basics of CMAC NN

The CMAC network functions as a static associative memory that models the nonlinear mapping between the mossy fiber inputs and the Purkinje cell outputs of the human cerebellum. The massive mesh of granule cell encoders in the cerebellum corresponds to an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells form an array of modifiable synaptic weights that motivates the grid-like CMAC computing structure. In the human cerebellum, these modifiable synaptic weights are linearly combined by the Purkinje cells to form the cerebellar output. In CMAC, the network output is likewise computed by aggregating the memory contents of the active computing cells.

Fig. 2 illustrates the computational mechanisms of the CMAC NN [2]. The CMAC network is essentially implemented as a multidimensional memory array, in which an input vector acts as the address decoder to access the respective memory cells containing the adjustable weight parameters to compute the corresponding output. CMAC learns the correct output response to each input vector by modifying the contents of the selected memory locations. The learning mechanism adopted in the CMAC network is based on error correction. For each input vector, the difference (error) between the CMAC output and the known desired response is computed and the

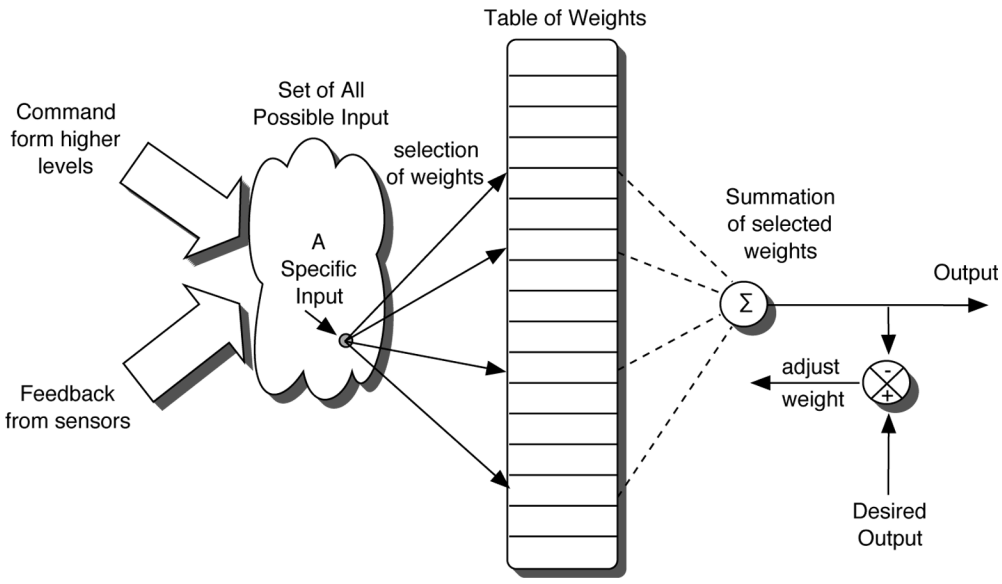


Fig. 2. Schematic diagram of the CMAC NN (adapted from [2]).

weight values of the selected memory cells in the network are adjusted accordingly.

B. Single-Layered Implementation of the CMAC Network

In the original implementation of the CMAC network [47], the memory cells of the network are divided into layers. The number of layers in a CMAC network is determined by the number of *quantization functions* defined. That is, one quantization function corresponds to one layer. Fig. 3 depicts an example of a two-input CMAC network with four quantization functions in each of the input dimensions. The resultant 2-D grid in Fig. 3 corresponds to the input space of the CMAC network that is used to learn the associative mapping patterns for 256 input–output (I/O) vector combinations. The quantization functions of the network are defined as follows.
Along dimension S_1

$$\begin{aligned} {}^{S_1}Q_1 &\rightarrow \{A, B, C, D\} \\ {}^{S_1}Q_2 &\rightarrow \{E, F, G, H, I\} \\ {}^{S_1}Q_3 &\rightarrow \{J, K, L, M, N\} \\ {}^{S_1}Q_4 &\rightarrow \{O, P, Q, R, S\} \end{aligned} \tag{1}$$

Along dimension S_2

$$\begin{aligned} {}^{S_2}Q_1 &\rightarrow \{a, b, c, d\} \\ {}^{S_2}Q_2 &\rightarrow \{e, f, g, h, i\} \\ {}^{S_2}Q_3 &\rightarrow \{j, k, l, m, n\} \\ {}^{S_2}Q_4 &\rightarrow \{o, p, q, r, s\} \end{aligned} \tag{2}$$

where ${}^{S_i}Q_j$ denotes the quantization function for the jth layer of the ith dimension and $\{\}$ denotes the set of quantization levels. Each input vector selects *one* memory cell from a layer. In the CMAC of Fig. 3, the input vector of (6,6) selects a total of four memory cells (one from each layer). That is, the $C1, C2, C3$, and $C4$ cells that correspond to the quantization points of Bb, Gg, Ll, and Pp, respectively. The output of the CMAC network is subsequently computed by the linear combination of the memory contents of the selected cells.

However, the multilayered structure of the CMAC network often renders the network operations difficult to comprehend. Moreover, in such an implementation, extensive layers of overlapping computing cells are required to produce a smooth output. The optimization of the memory allocation process (nonuniform quantization) of a multilayered, multi-input CMAC NN is not only tedious, but can also be computationally expensive, especially for high-dimensional input problems that require extensive layers of computing cells to achieve the desired output resolution or accuracy. This is because it is difficult to manipulate the distribution of the memory cells in the individual layers as the computation of all the overlapping layers are intertwined and tightly coupled to produce the CMAC output. Therefore, this paper proposes a single-layered CMAC implementation for the optimization of the memory allocation procedure based on hierarchical clustering. In such a computing structure, memory allocation or the distribution of the memory cells becomes manageable as there is only one layer of computing cells.

Fig. 4 illustrates such a single-layered perspective of a two-input CMAC network consisting of 64 memory cells. The 2-D computing grid corresponds to the memory space of the CMAC network. In Fig. 4, each of the input dimensions is uniformly *quantized* into eight discrete quantization steps (or levels) for direct comparison with the multilayered CMAC of Fig. 3. Similar to the multilayered CMAC, the input vector to this network is quantized to the corresponding level for each input dimension to obtain the index address of the winner memory cell. Smoothing of the computed output is achieved by a *neighborhood activation* of the computing cells. The activated neighborhood for the input vector (6,6) to the single-layered CMAC implementation is depicted in Fig. 4. That is, to ensure the continuity in the output surface, each input vector selects a cluster of memory cells or neurons that is centered at the winner neuron (see shaded square of Fig. 4).

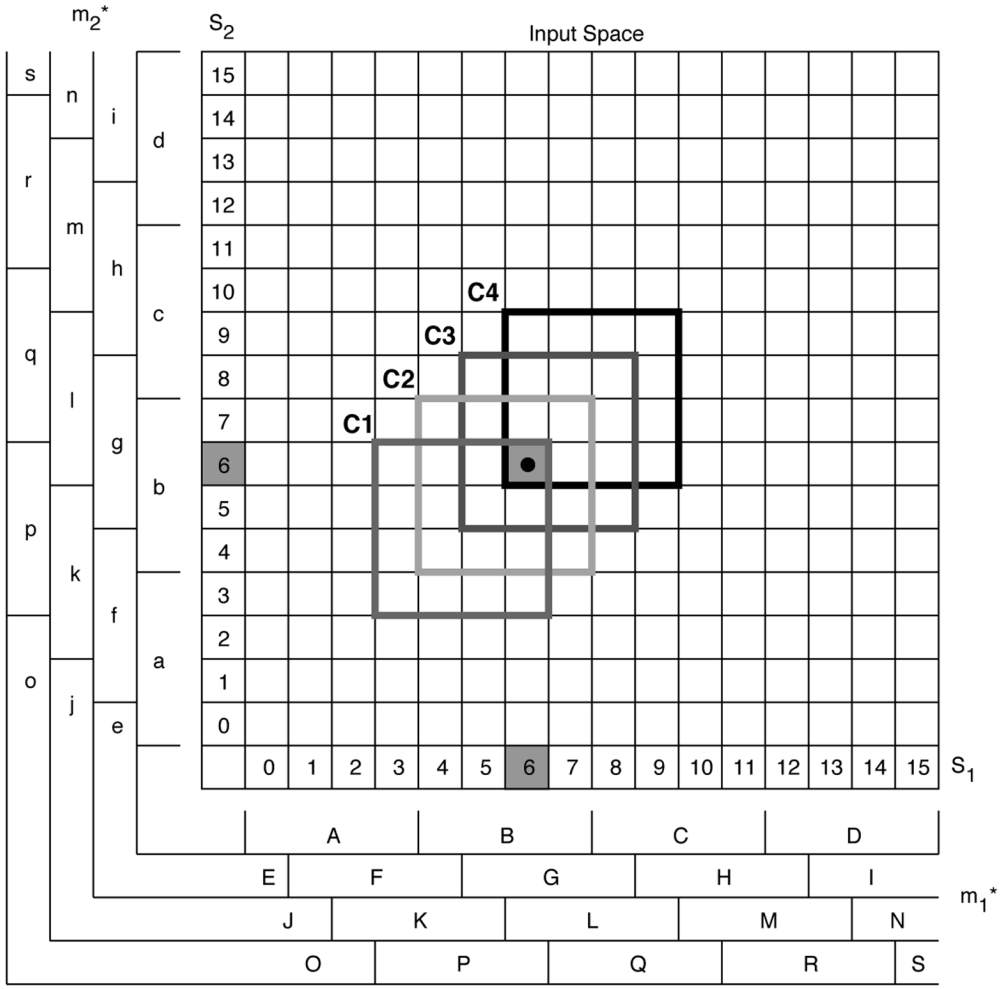


Fig. 3. An example of a 2D CMAC network (m_1^* refers to the set of quantization functions along the S_1 dimension and m_2^* refers to the set of quantization functions along the S_2 dimension).

The conceptual similarities between the proposed single-layered model and the original multilayered implementation of the CMAC network can be examined from their respective modeling principles. The layered cell activations in the original CMAC network contributed to the following two significant computational objectives: 1) smoothing of the computed output and 2) activating similar or highly correlated computing cells in the I/O associative space. These two modeling principles are similarly conserved in the single-layered model of the CMAC network via the introduction of a neighborhood-based computational process. The activation of the neighboring cells in the input space of the single-layered CMAC corresponds to the simultaneous activation of the highly correlated cells in its multilayered counterpart. This contributes to the smoothing of the computed output since the neighborhood-based activation process results in continuity of the network output. This activation process will be further discussed in Section IV.

C. Nonuniform Quantization CMAC Variants

Several attempts to address the uniform quantization of CMAC can be found in the literature. In general, they can be classified into two main approaches based on the type of nonuniformity introduced. The first approach uses layers/hierarchy of multiresolution CMAC networks to achieve nonlinearity in memory storage allocation. Originally proposed by Moody [25], a CMAC is trained using coarsely quantized (low-resolution) inputs. For regions where the error is large, the decision to add another CMAC with higher resolution is made. The expansion process continues until the error is reduced to an acceptable level. Some variations and applications of this type of nonuniform CMAC have also been proposed in [17] and [22]–[24]. While these methods can effectively capture the general trend as well as fine details in the overall target function to be learned, the total number of memory cells needed to achieve a particular performance is not known in advance, making any hardware implementation awkward and difficult.

The second approach uses a layer of quantization decision function as the address decoder/mapper between the input to the network and the index to the CMAC. For example, [27] employs competitive learning to find the topological structure of the input samples in order to determine the quantization decision function. In [26], the use of gray relational analysis as the adaptive quantization technique for the input layer is proposed. Another example is found in [28] which makes use of the minimization output error criteria in adjusting the quantization step size. While these methods aim to effectively allocate a predefined amount of memory cells, the quantization decision function of these methods are based on the derivative

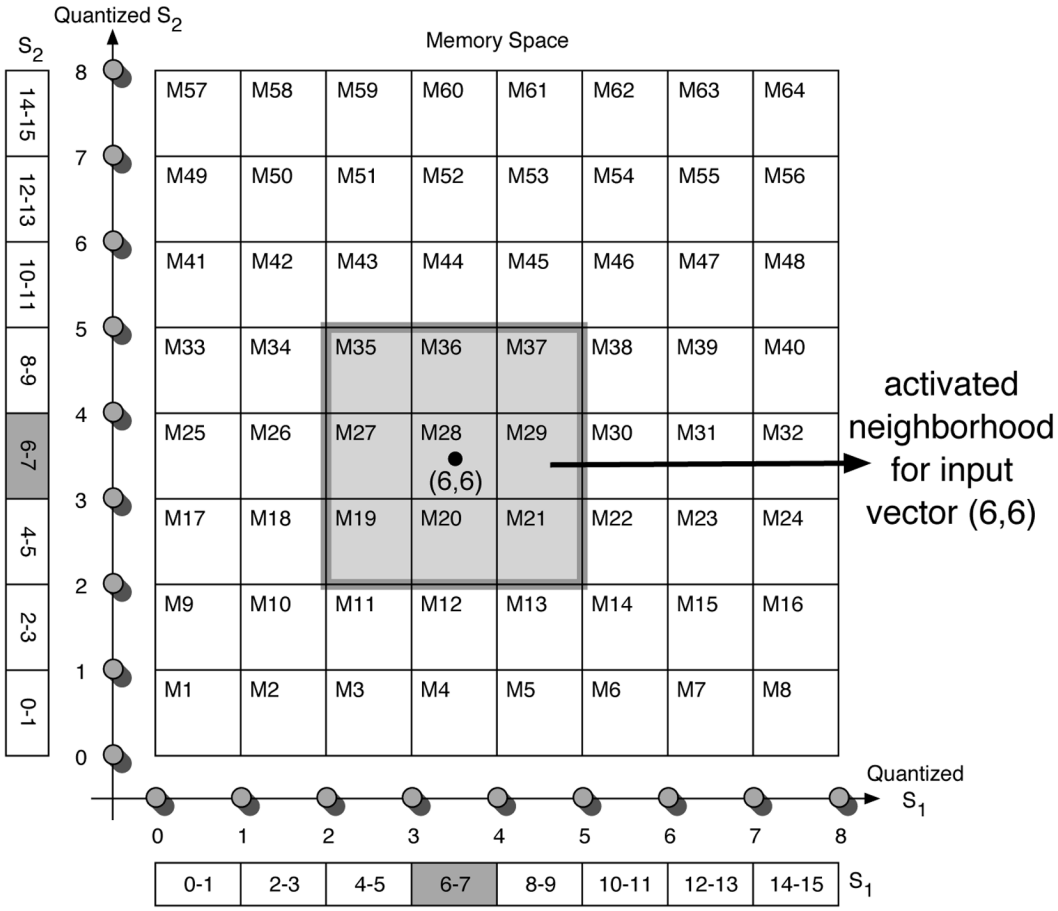


Fig. 4. Single-layer perspective of the 2-D CMAC network example shown in Fig. 3.

of the target function to be learned, which is often difficult to obtain or not known in advance. Furthermore, for these methods, the storage efficiency achieved is at the cost of significantly higher computational complexity. On the other hand, the adaptive quantization approaches proposed in [29] and [30] do not require the target derivative information. In [30], a data clustering algorithm is employed to find the centers of the partitions in the CMAC layers. In [29], the Shannon's entropy measure is used to adaptively determine the information distribution of the training data. However, since entropy is measurable only for discrete/nominal inputs, this approach is more suited for classification problems.

Other CMAC variants such as the kernel CMACs [48]–[50] and the fuzzy CMACs [51], [52] do not employ quantization-based addressing schemes. In kernel CMAC, kernel functions such as B-spline kernels are used to derive the receptive fields of the network. While this method is able to reduce the memory requirement, the choice of kernel function is still an open problem. Moreover, the complexity of the resultant CMAC depends on the kernel function used. Fuzzy CMACs, on the other hand, employ input fuzzification methods to remove the sharp quantization boundary of the original CMAC network. The use of fuzzy inference schemes such as in [52] also enables the extraction of fuzzy rules from the trained network. While these variants offered the computational interpretability missing in the black-box CMAC model, there is a significant increase in the computational complexity of the hybrid network without a comparable performance gain.

IV. HCAQ-CMAC

The operational principle of the uniform quantization scheme in the CMAC network results in a constant output resolution throughout the entire input space. On the other hand, many meaningful real-life applications are generally *heteroskedastic* in nature, where there exists several distinct sets of trends that are significant for the precise characterization of the given problem. That is, due to the underlying characteristics and dynamics of the training data, certain regions of the input space tend to contain more information than the rest. Therefore, drawing inspiration from the neurophysiological studies on programmed cell death and competitive survival of neurons, as well as from the experience-driven brain development and plasticity, we propose a novel CMAC architecture named the HCAQ-CMAC as an alternative to realize nonuniform quantization. Unlike the existing approaches outlined in Section III-C, HCAQ-CMAC efficiently allocates the number of available memory cells based on the characteristics/information contents of the I/O mappings of the training data. The neuronal competition for survival in the human brain is emulated by the hierarchical clustering technique, where similar data points are clustered together to eliminate redundancy in data representation. This similarity is measured by the observed degree of variations of the target output. In the HCAQ-CMAC network, memory efficiency is achieved by allocating more memory cells to the input regions where rapid fluctuations of the output values are observed, and less memory cells to

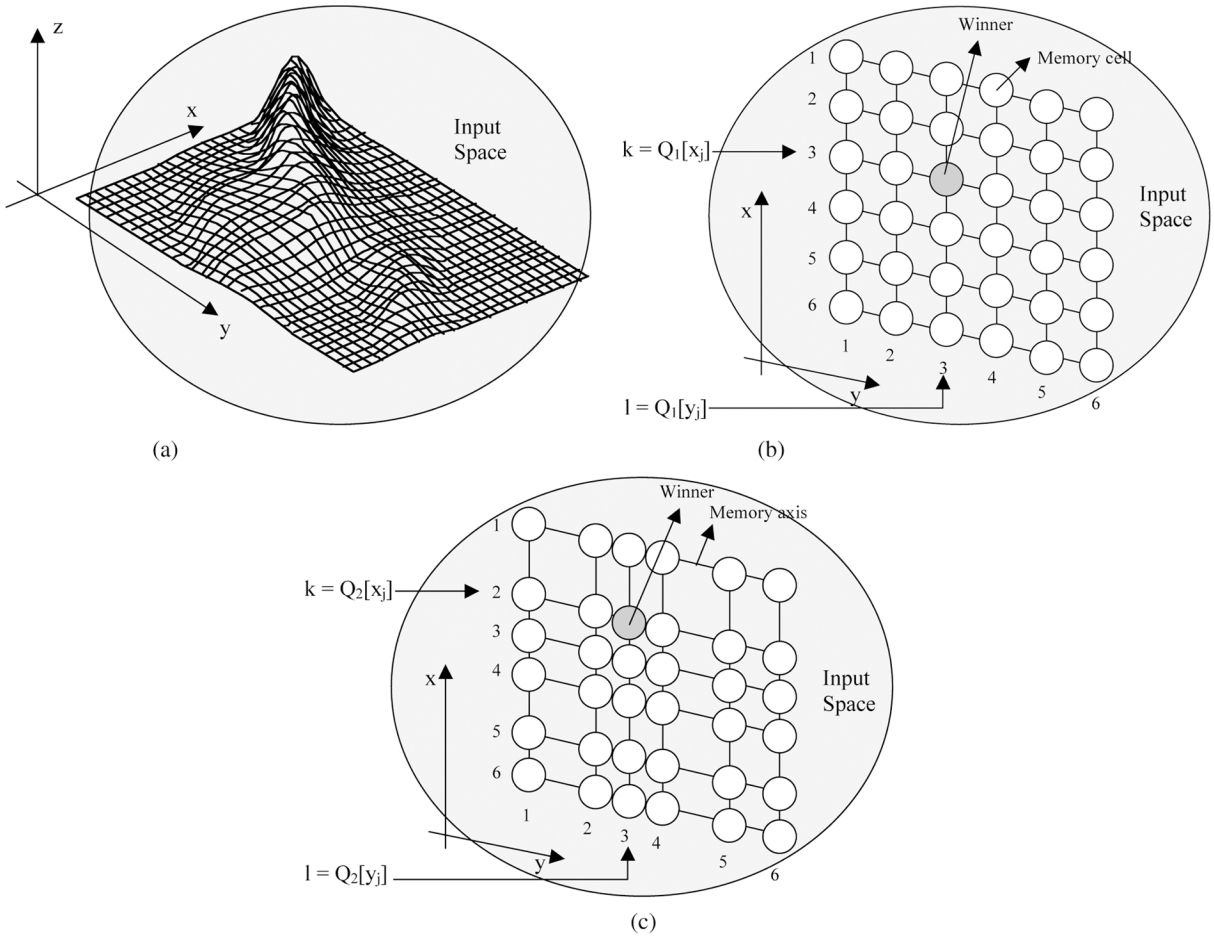


Fig. 5. Comparison of CMAC and HCAQ-CMAC memory structure for a particular target output surface. (a) I/O characteristic. (b) The 2-D CMAC memory structure. (c) The 2-D HCAQ-CMAC memory structure.

regions with relatively unchanged output values. As a result, a finer quantization level is obtained for the regions of the input space which contain more information. This nonuniform quantization process is illustrated in Fig. 5. Fig. 5(a) shows an example of a target output surface of a two-input problem to be modeled. Fig. 5(b) depicts the corresponding uniformly quantized CMAC memory structure and Fig. 5(c) illustrates the nonuniform HCAQ-CMAC for the same target output surface.

A. HCAQ-CMAC Network Architecture

Neurophysiological studies have established that the precise wiring of the adult human brain is not fully developed at birth [4]. Instead, there are two overlapping stages in the development of the human central nervous system. The embryonic stage of this process encompasses the formation of the basic architecture of the nervous system, in which coarse connection pattern emerges as a result of the genesis and death of the brain cells during prenatal development. Subsequently, in the postnatal development stage, the initial architecture is refined and extraneous synaptic connections are pruned throughout an individual's life-span by repeated exposures to various activity-dependent experiences. These processes constitute a selective allocation of neurons in the human brain and is incorporated to the HCAQ-CMAC memory allocation procedure as a mechanism for nonuniform quantization.

The HCAQ-CMAC memory allocation procedure is inspired by the biological development of the human central nervous system where neural cell death plays an integral part in the refinement process of the brain's neuronal organization. In HCAQ-CMAC, the available memory cells are distributed based on the observed characteristics of the training data, which are defined by the data distribution in the input space as well as the variation of the target output value. For this reason, the HCAQ-CMAC architecture presented in this paper is a multiple-input–single-output (MISO) CMAC variant. This is because in a multiple-output domain, the variation of each output variable may not be correlated to one another. Thus, it is difficult to formulate a quantization function that is optimal for all output dimensions. A multiple-input–multiple-output (MIMO) problem can instead be modeled by a combination of several MISO systems.

In HCAQ-CMAC, the nonuniform quantization process and the subsequent memory allocation procedure are performed on a per-dimension basis. There are the following two stages in the operation of the HCAQ-CMAC network: 1) network initialization stage and 2) network learning stage. The purpose of the network initialization stage is to define the quantization function at each of the input dimensions, while the network learning stage is to learn the memory contents of the HCAQ-CMAC network. The agglomerative hierarchical clustering technique [53]

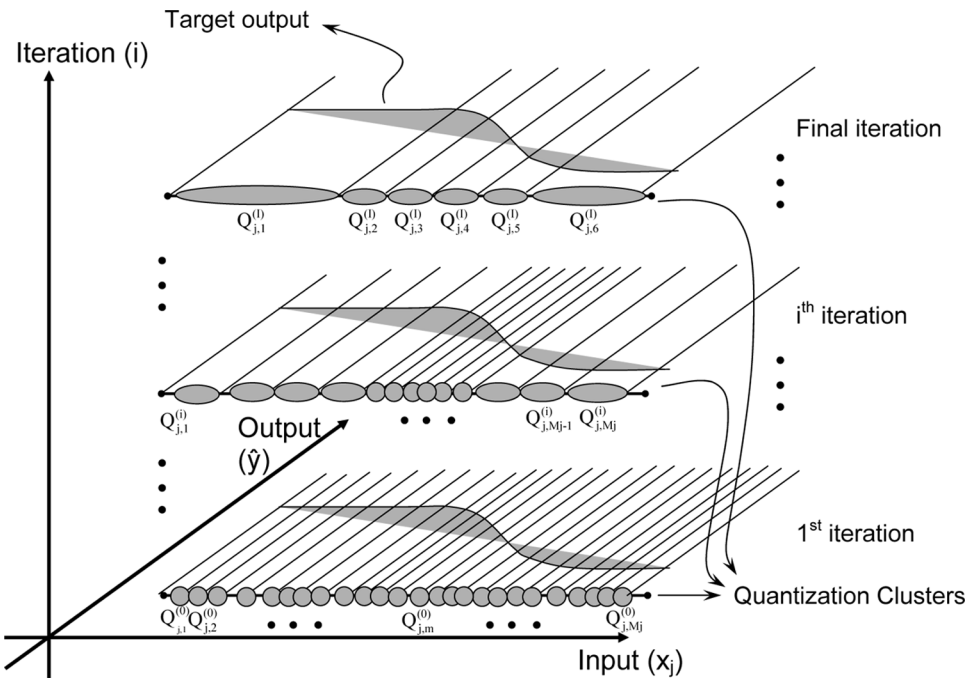


Fig. 6. Illustration of the HCAQ-CMAC nonuniform quantization process for an arbitrary input dimension j where $\hat{M} = 6$.

is employed at each dimension to identify the optimal quantization decision function in each of the input dimensions. A *quantization cluster* is defined as the span of a memory quantization level in a particular input dimension. Starting with the initial set of quantization clusters in a particular input dimension, two clusters with the smallest merging cost are combined in each iteration of the hierarchical clustering process until the number of quantization clusters is equal to the number of available (predefined) memory space in the respective input dimension (see Fig. 6).

Let J denote the total number of input dimensions for a given problem. Assume that a training data set of $\mathbf{U} = \{(\mathbf{X}_1, \hat{Y}_1), (\mathbf{X}_2, \hat{Y}_2), \ldots, (\mathbf{X}_s, \hat{Y}_s), \ldots, (\mathbf{X}_S, \hat{Y}_S)\}$ is used to train the HCAQ-CMAC network, where $\mathbf{X}_s = [x_{s,1}\ x_{s,2}\ \ldots\ x_{s,J}]^T$ denotes the sth input vector to the network and $\hat{Y}_s$ denotes the expected scalar output of HCAQ-CMAC. Let $\hat{M}$ denote the total number of available memory cells per dimension and $\mathbf{Q}_j^{(i)} = \{Q_{j,1}^{(i)}, Q_{j,2}^{(i)}, \ldots, Q_{j,m}^{(i)}, \ldots, Q_{j,M_j}^{(i)}\}$ denote a set of M_j quantization clusters in the jth input dimension at the ith iteration of the hierarchical clustering process. The HCAQ-CMAC memory allocation process is described as follows:

Step 1) Perform data preparation. For each input dimension $j \in \{1 \cdots J\}$, the input training samples $x_{s,j}$'s are sorted in ascending order such that $\{x_{1,j}, x_{2,j}, \ldots, x_{s,j}, \ldots, x_{S,j}\}$, where $x_{s,j} \leq x_{s+1,j}$.

Step 2) Define the initial set of quantization clusters. Let $Q_{j,m}^{(i)}$ denote the mth quantization cluster in the jth input dimension at the ith iteration of the hierarchical clustering process, and $n_{Q_{j,m}^{(i)}}$ as the number of data points in $Q_{j,m}^{(i)}$. Then, the quantization cluster $Q_{j,m}^{(i)}$ is defined as the set of I/O data pairs such that $Q_{j,m}^{(i)} = \{(x_{j,m\_1}^{'(i)}, y_{j,m\_1}^{'(i)}), (x_{j,m\_2}^{'(i)}, y_{j,m\_2}^{'(i)}), \ldots, (x_{j,m\_n_{Q_{j,m}}}^{'(i)}, y_{j,m\_n_{Q_{j,m}}}^{'(i)})\}$. For each input dimension j, where $j \in \{1 \cdots J\}$, the initial set of quantization clusters $\mathbf{Q}_j^{(0)}$ is derived from the input training data points $x_{s,j}, s \in \{1 \cdots S\}$ in the jth input dimension. Each *distinct* input value $x_{s,j}$ from the training set constitutes *one* quantization cluster in the jth dimension. Training data points with the same input value are combined together as a cluster such that

$$Q_{j,m}^{(0)} = \left\{ \left(x_{j,m\_1}^{'(0)}, y_{j,m\_1}^{'(0)}\right), \left(x_{j,m\_2}^{'(0)}, y_{j,m\_2}^{'(0)}\right), \ldots, \left(x_{j,m\_n_{Q_{j,m}}}^{'(0)}, y_{j,m\_n_{Q_{j,m}}}^{'(0)}\right) \right\} \quad (3)$$

where $x_{j,m\_1}^{'(0)} = x_{j,m\_2}^{'(0)} = \cdots = x_{j,m\_n_{Q_{j,m}}}^{'(0)} = x_{j,m}^{'(0)}$. Each quantization cluster $Q_{j,m}^{(i)}$ is defined by a characteristic value $V_{j,m}^{(i)}$ and its centroid $P_{j,m}^{(i)}$. The characteristic value $V_{j,m}^{(i)}$ of a quantization cluster $Q_{j,m}^{(i)}$ at the ith iteration of the hierarchical clustering process is computed as the mean of the output values $y_{j,m\_k}^{'(i)}$ of the input points $x_{j,m\_k}^{'(i)}$ and is described by

$$V_{j,m}^{(i)} = \frac{\sum_k y_{j,m\_k}^{'(i)}}{n_{Q_{j,m}}}, \qquad k \in \{1 \cdots n_{Q_{j,m}}\} \quad (4)$$

where $n_{Q_{j,m}}$ is the total number of data points in the quantization cluster $Q_{j,m}^{(i)}$ at the ith iteration. $P_{j,m}^{(i)}$, on the other hand, denotes the centroid of the quantization cluster $Q_{j,m}^{(i)}$ and is described by

$$P_{j,m}^{(i)} = \frac{\max\left(x_{j,m\_k}^{'(i)}\right) - \min\left(x_{j,m\_k}^{'(i)}\right)}{2}, \quad k \in \{1 \cdots n_{Q_{j,m}}\}. \quad (5)$$

With respect to the input dimension j, the initial set of quantization clusters is defined as $\mathbf{Q}_j^{(0)} = \{Q_{j,1}^{(0)}, Q_{j,2}^{(0)}, \ldots, Q_{j,M_j}^{(0)}\}$ where $V_{j,m}^{(0)} = (\sum_k y_{j,m\_k}^{,(0)})/n_{Q_{j,m}}$, and $k \in \{1 \cdots n_{Q_{j,m}}\}$ and $P_{j,m}^{(0)} = x_{j,m}^{,(0)}$ (since $x_{j,m\_1}^{,(0)} = x_{j,m\_2}^{,(0)} = \cdots = x_{j,m\_n_{Q_{j,m}}}^{,(0)} = x_{j,m}^{,(0)}$).

Step 3) Merge similar quantization clusters iteratively. The clusters in the initial set of quantization clusters $\mathbf{Q}_j^{(0)}$ are iteratively merged until the number of quantization clusters in the jth dimension M_j is equal to the number of predefined available memory cells, i.e., $M_j = \hat{M}$. Only the two most similar adjacent clusters can be merged in each iteration. The cluster-merging decision is based on a *cost function*. The merging cost function is defined as the weighted combination of the distances between the characteristic values and the centroids of two adjacent clusters, and is described mathematically in

$$\mathfrak{F}\left(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)}\right) = \beta_1 \left(\left\|V_{j,m_1}^{(i)} - V_{j,m_2}^{(i)}\right\|\right) + \beta_2 \left(\left\|P_{j,m_1}^{(i)} - P_{j,m_2}^{(i)}\right\|\right) \quad (6)$$

$$\beta_1 + \beta_2 = 1.0 \quad (7)$$

where $\mathfrak{F}(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)})$ is the merging cost of the two adjacent clusters $Q_{j,m_1}^{(i)}$ and $Q_{j,m_2}^{(i)}$ (i.e., $m_1, m_2 \in \{1 \cdots M_j\}$ and $m_2 = m_1 + 1$) at the ith iteration, and β_1 and β_2 are user-defined parameters. The parameters β_1 and β_2 weight the respective importance of the measured differences in the output (characteristic values) and the input (quantization points) dimensions as the total cost of merging two adjacent clusters. The weighting parameter β_1 is concerned with the similarity of the outputs of the two clusters. β_2, on the other hand, controls the importance of the similarity of the inputs in the two clusters. As such, the selection of β_1 and β_2 parameters generally varies greatly with different applications and may be guided by the relevant prior knowledge about the applications or the training data. An application with slow changing input but fast changing output may require a bigger β_1 than β_2 and vice-versa.

In each iteration, the two adjacent clusters with the smallest merging cost are combined as in

$$Q_{j,m'}^{(i+1)} = Q_{j,\tilde{m}_1}^{(i)} \cup Q_{j,\tilde{m}_2}^{(i)}$$

iff

$$\mathfrak{F}\left(Q_{j,\tilde{m}_1}^{(i)}, Q_{j,\tilde{m}_2}^{(i)}\right) = \min_{m_1, m_2 \in \{1 \cdots M_j\} \text{ and } m_2 = m_1 + 1} \mathfrak{F}\left(Q_{j,m_1}^{(i)}, Q_{j,m_2}^{(i)}\right). \quad (8)$$

The characteristic value $V_{j,m'}^{(i+1)}$ and the centroid $P_{j,m'}^{(i+1)}$ of the merged cluster $Q_{j,m'}^{(i+1)}$ are recomputed using (4) and (5). The cluster-merging process continues until the number of clusters in the input dimension j reaches the predefined memory size $\hat{M}$ (see Fig. 6). This is analogical to the activity-dependent pruning of the extraneous synaptic connections in the human brain. Weak (or nonactive) neurons are eliminated and their functions annexed by the winning (or more active) neurons. In the HCAQ-CMAC, similar clusters are merged and represented by larger/expanded clusters to reduce data redundancy. Specifically, HCAQ-CMAC allocates more memory cells to the densely data-populated areas with higher degrees of output variation.

Step 4) Construct the quantization decision function. A set of $\hat{M}$ quantization clusters is obtained at the end of the hierarchical clustering process for each input dimension j. Let I denote the last cluster-merging iteration for input j. Thus, the final set of quantization clusters for input j is defined as $\mathbf{Q}_j^{(I)} = \{Q_{j,1}^{(I)}, Q_{j,2}^{(I)}, \ldots, Q_{j,\hat{M}}^{(I)}\}$. Subsequently, the quantization decision function in the jth input dimension $\mathbf{Q}_j[\cdot]$ is determined from $\mathbf{Q}_j^{(I)}$, as described by

$$\mathbf{Q}_j[\cdot] \rightarrow \left\{P_{j,1}^{(I)}, P_{j,2}^{(I)}, \ldots, P_{j,\hat{M}}^{(I)}\right\} \quad (9)$$

where $\mathbf{Q}_j[\cdot]$ denotes the quantization mapping function in the jth input dimension. $P_{j,m}^{(I)}$ is the centroid of the mth quantization cluster of the jth input dimension after the cluster-merging process and $\hat{M}$ is the number of predefined memory cells in each dimension. The quantization decision points derived for each input dimension subsequently form the memory axes of the HCAQ-CMAC network and define its overall computing structure.

B. HCAQ-CMAC Operating Principles

The HCAQ-CMAC network learns a correct response to an input vector by modifying the contents of the selected memory cells. Let I be the maximum number of training iterations. The Widrow–Hoff learning rule [54] is adopted and the HCAQ-CMAC memory learning process for the sth training sample is described as follows.

Step 1) Determine the winner neuron for input $\mathbf{X}_s$ at the ith iteration, where $i \in \{1 \cdots I\}$. For each input $\mathbf{X}_s$, the index $\bar{\mathbf{X}}_s$ of the winner neuron in HCAQ-CMAC is computed via the quantization mapping functions $\mathbf{Q}[\cdot]$. That is, given $\mathbf{X}_s = [x_{s,1}\ x_{s,2}\ \cdots\ x_{s,j}\ \cdots\ x_{s,J}]^T$, the winner neuron $\bar{\mathbf{X}}_s$ is as in

$$\bar{\mathbf{X}}_s = \mathbf{Q}[\mathbf{X_s}] = [\mathbf{Q}_1[x_{s,1}]\ \mathbf{Q}_2[x_{s,2}]\ \cdots\ \mathbf{Q}_j[x_{s,j}]\ \cdots\ \mathbf{Q}_J[x_{s,J}]]^T \quad (10)$$

where $\bar{\mathbf{X}}_s$ denotes the quantized input $\mathbf{X}_s$ and J is the number of input dimensions.

Step 2) Retrieve the network output. The output of the HCAQ-CMAC network to the input $\mathbf{X}_s$ at the ith

training iteration is the memory content at the location $\bar{\mathbf{X}}_s$. This is described by

$$Y_s^{(i)} = \mathbf{Z}_{\bar{\mathbf{X}}_s} \tag{11}$$

where $Y_s^{(i)}$ denotes the HCAQ-CMAC output for the input $\mathbf{X}_s$ during the ith training iteration and $\mathbf{Z}$ is the HCAQ-CMAC hypercube memory array.

Step 3) Compute the network output error. The learning error $\mathrm{Err}_s^{(i)}$ corresponding to the input $\mathbf{X}_s$ at the ith training iteration is defined as the difference between the network output $Y_s^{(i)}$ and the expected output $\hat{Y}_s$ is given in

$$\begin{aligned}\mathrm{Err}_s^{(i)} &= \hat{Y}_s - Y_s^{(i)} \\ &= \hat{Y}_s - \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)}.\end{aligned} \tag{12}$$

Step 4) Update the HCAQ-CMAC memory. The update equation for the activated cell at index $\bar{\mathbf{X}}_s$ is given by

$$\mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i+1)} = \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} + \Delta\mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} \tag{13}$$

$$\Delta\mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)} = \alpha\mathrm{Err}_s^{(i)} = \alpha\left(\hat{Y}_s - \mathbf{Z}_{\bar{\mathbf{X}}_s}^{(i)}\right) \tag{14}$$

where α denotes the learning constant.

During the testing phase of the HCAQ-CMAC network, a neighborhood-based activation of the network cells is employed to smoothen the computed output. Given an input stimulus $\mathbf{X}_s = [x_{s,1}\ x_{s,2}\ \cdots\ x_{s,j}\cdots\ x_{s,J}]^T$ to the HCAQ-CMAC network, the network output during the testing phase is derived as follows.

Step 1) Determine the region of activation. The computed output of the HCAQ-CMAC network corresponding to an input stimulus $\mathbf{X}_s$ is defined as the mean of the memory contents (values) of the activated cells in the neighborhood vicinity of $\mathbf{X}_s$. The neighborhood of $\mathbf{X}_s$ is defined by a neighborhood constant N, which determines the relative size of the neighborhood with respect to the input domain. For an input stimulus $\mathbf{X}_s$, its activation neighborhood is defined by

$$lb_{s,j} = x_{s,j} - 0.5 \cdot N \cdot \mathrm{range}_{s,j} \tag{15}$$

$$rb_{s,j} = x_{s,j} + 0.5 \cdot N \cdot \mathrm{range}_{s,j} \tag{16}$$

where $j \in \{1, 2, \ldots, J\}$ denotes the jth input dimension, N is the neighborhood constant, $\mathrm{range}_{s,j}$ is the input domain for the jth dimension, $lb_{s,j}$ is the left boundary of the neighborhood in the jth dimension, and $rb_{s,j}$ is the right boundary of the neighborhood in the jth dimension. Consequently, the memory cells within the neighborhood constitute the set of activated computing cells for the input stimulus $\mathbf{X}_s$. The size of the neighborhood affects the accuracy of the computed HCAQ-CMAC output. The larger the neighborhood size, the more generalized is the output of the HCAQ-CMAC network. Conversely, a smaller neighborhood size results in a more accurate output computation. Therefore, a larger neighborhood size is suitable for a data set that is sparse in the input space as this increases the generalization ability of the HCAQ-CMAC network. A smaller neighborhood size, on the other hand, is suitable for a compact data set so as to produce more accurate results.

Step 2) Compute the HCAQ-CMAC output. The output of the HCAQ-CMAC network with respect to the input $\mathbf{X}_s$ is defined by

$$\hat{Y}_s = \frac{\sum_{\mathbf{k}\in\mathbf{K}_s}\mathbf{Z}_k}{n_{\mathbf{K}_s}} \tag{17}$$

where $\mathbf{K}_s$ denotes the set of indexes of the activated neighborhood cells corresponding to the input $\mathbf{X}_s$, $\mathbf{Z}_k$ is the memory content of the activated cell with index k, $n_{\mathbf{K}_s}$ is the cardinality of $\mathbf{K}_s$, and $\hat{Y}_s$ is the output of HCAQ-CMAC with respect to the input stimulus $\mathbf{X}_s$.

As a computational model of the human cerebellum, the proposed HCAQ-CMAC network possesses characteristics analogical to the neurobiological and neurophysiological aspects of its biological counterpart. Appendix A lists the neural correlates between the human cerebellum and the HCAQ-CMAC network.

V. HCAQ-CMAC Learning Convergence

This section presents the mathematical proof of the learning convergence of the proposed HCAQ-CMAC network. Fig. 7 depicts an example of the memory surface $\mathbf{Z}$ of a two-input HCAQ-CMAC network. With respect to Fig. 7, the quantization points along the X_1 dimension are $\{P_{1,1}, P_{1,2}, P_{1,3}, \ldots P_{1,\hat{M}}\}$ and along the X_2 dimension are $\{P_{2,1}, P_{2,2}, P_{2,3}, \ldots P_{2,\hat{M}}\}$, respectively. $\mathbf{Z}_{(p1,p2)}$ denotes the network cell with the address index $(p1, p2)$.

A. Mathematical Perspective of the HCAQ-CMAC Network

The HCAQ-CMAC network employs a winner-take-all learning principle where each input training tuple accesses and modifies the memory content of *one* winner neuron. Each input vector to the network is quantized to the nearest quantization level in each dimension to identify the index of the winner neuron. The HCAQ-CMAC network output is derived from the winner network cell. Consequently, the network learning process is performed on this winner cell.

The conceptual memory surface $\mathbf{Z}$ of a multiple-input HCAQ-CMAC network can be expressed as a 1-D weight array $\mathbf{W}$. Fig. 8 illustrates the *linearization* of the conceptual memory surface $\mathbf{Z}$ to the physically implemented 1-D weight array $\mathbf{W}$ for a 2-D HCAQ-CMAC. With respect to the HCAQ-CMAC network, the computed output for the sth input vector (stimulus) $\mathbf{X}_s$ is defined in

$$Y_s^{(i)} = \mathbf{Z}_{\mathbf{Q}[\mathbf{X}_s]} \tag{18}$$

where i is the training iteration number, $\mathbf{Q}[\cdot]$ denotes the quantization mapping function of the HCAQ-CMAC network, and $\mathbf{Q}[\mathbf{X}_s]$ is the index to the winner neuron corresponding to the input $\mathbf{X}_s$.

The HCAQ-CMAC network presented here is a MISO system. Let the total number of memory cells in the

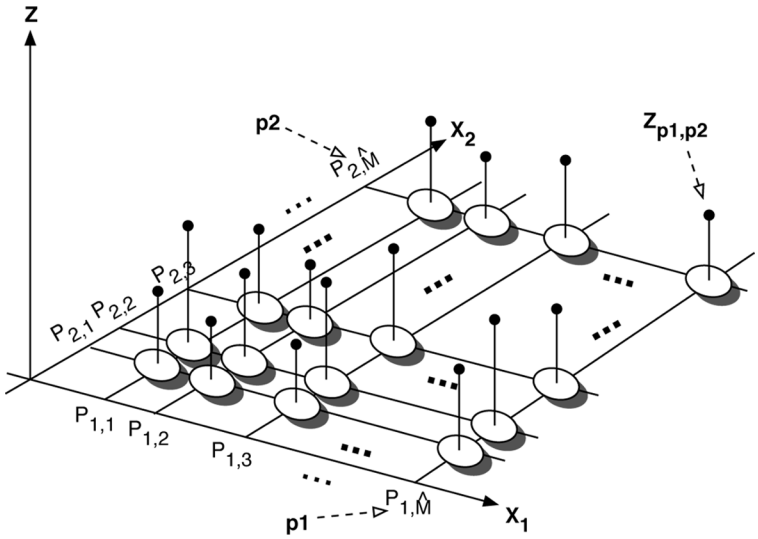


Fig. 7. Example of the two-input HCAQ-CMAC memory content.

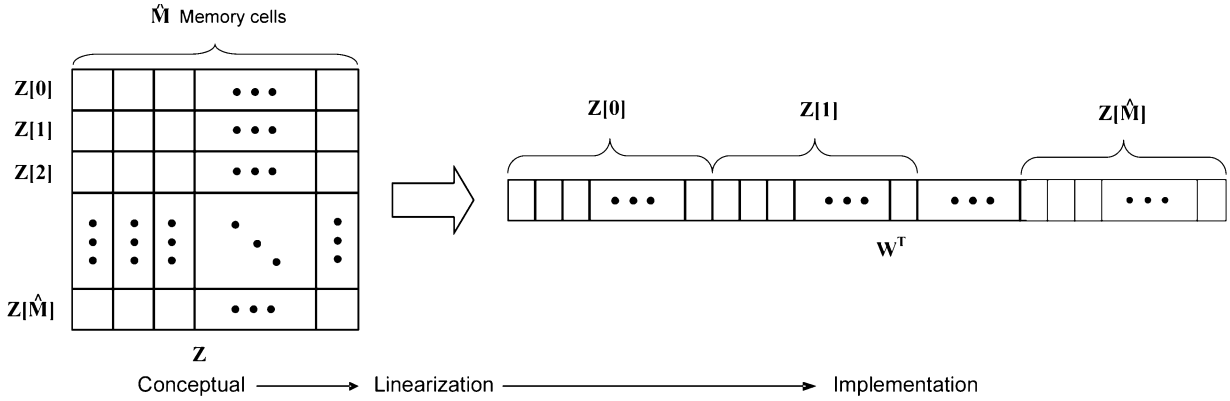


Fig. 8. The 2-D HCAQ-CMAC $\mathbf{Z} \rightarrow \mathbf{W}$ mapping.

HCAQ-CMAC network be $\hat{M}^J$ and the column vector $\mathbf{C}_s$ [see (19)] denote the activation mask of the HCAQ-CMAC memory cells with respect to the sth input training sample. That is

$$\mathbf{C}_s^T = \underbrace{[c_{s,1}\ c_{s,2}\ \cdots\ c_{s,\hat{M}^J}]}_{1\times\hat{M}^J \text{ array}} \tag{19}$$

$$c_{s,j} = \begin{cases} 1, & \text{if the } j\text{th memory cell is activated} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Note that the winner-take-all learning algorithm of the HCAQ-CMAC network implies that, for all $s \in \{1 \cdots S\}$, only one element of $\mathbf{C}_s$ is nonzero. The scalar output of the HCAQ-CMAC network can thus be formulated as a vector product described by

$$Y_s = \underbrace{[c_{s,1}\ c_{s,2}\ \cdots\ c_{s,\hat{M}^J}]}_{1\times\hat{M}^J \text{ array}} \mathbf{W}_s = \mathbf{C}_s^T \mathbf{W}_s \tag{21}$$

where $\mathbf{W}_s$ is the memory content of the entire HCAQ-CMAC network structure when the sth input training sample is presented.

The memory update equation of the HCAQ-CMAC network for the sth input training sample is subsequently defined as

$$\begin{aligned} \mathbf{W}_{s+1}^{(i)} &= \mathbf{W}_s^{(i)} + \underbrace{\Delta\mathbf{W}_s^{(i)}}_{\alpha\times\text{local error}} \\ &= \mathbf{W}_s^{(i)} + \alpha \overbrace{\mathbf{C}_s \underbrace{\left\{\hat{Y}_s - \mathbf{C}_s^T \mathbf{W}_s^{(i)}\right\}}_{\text{local error}}}^{\text{learning error}} \end{aligned} \tag{22}$$

where

- $\mathbf{W}_{s+1}^{(i)}$ the memory content of the entire HCAQ-CMAC network structure when the $(s+1)$th training sample is presented in the ith training iteration;
- α the learning constant;
- $\mathbf{C}_s$ the activation mask of the HCAQ-CMAC memory cells;
- $\hat{Y}_s$ the desired (expected) HCAQ-CMAC output for the sth input training vector.

The difference of the HCAQ-CMAC memory contents between two successive iterations for the sth input training sample (denoted as $\mathbf{Dw}_s^{(i)}$) is, therefore, defined as

$$\begin{aligned}\mathbf{Dw}_s^{(i)} &= \mathbf{W}_s^{(i+1)} - \mathbf{W}_s^{(i)} \\ &= \underbrace{\mathbf{W}_{s-1}^{(i+1)} + \Delta\mathbf{W}_{s-1}^{(i+1)}}_{\mathbf{W}_s^{(i+1)}} - \underbrace{\left(\mathbf{W}_{s-1}^{(i)} + \Delta\mathbf{W}_{s-1}^{(i)}\right)}_{\mathbf{W}_s^{(i)}} \\ &= \underbrace{\mathbf{Dw}_{s-1}^{(i)}}_{\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)}} + \underbrace{\alpha\mathbf{C}_{s-1}\left\{\hat{Y}_{s-1} - \mathbf{C}_{s-1}^T\mathbf{W}_{s-1}^{(i+1)}\right\}}_{\Delta\mathbf{W}_{s-1}^{(i+1)}} \\ &\quad - \underbrace{\alpha\mathbf{C}_{s-1}\left\{\hat{Y}_{s-1} - \mathbf{C}_{s-1}^T\mathbf{W}_{s-1}^{(i)}\right\}}_{\Delta\mathbf{W}_{s-1}^{(i)}} \\ &= \mathbf{Dw}_{s-1}^{(i)} - \alpha\mathbf{C}_{s-1}\mathbf{C}_{s-1}^T\underbrace{\left(\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)}\right)}_{\mathbf{Dw}_{s-1}^{(i)}} \\ &= \left(\mathbf{I} - \alpha\mathbf{C}_{s-1}\mathbf{C}_{s-1}^T\right)\mathbf{Dw}_{s-1}^{(i)}. \end{aligned} \tag{23}$$

Note that the activation mask $\mathbf{C}_s$ is a constant for an arbitrary input training sample s across different training iterations. This is because the HCAQ-CMAC network structure is static after the structural learning phase.

Following (23), the delta memory contents for a sequence of training data $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_s, \ldots, \mathbf{X}_S)$ is described by

$$\mathbf{E}_s \equiv \left(\mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T\right) \tag{24}$$

and

$$\mathbf{Dw}^{(i)} \equiv \left[\mathbf{Dw}_1^{(i)}\ \mathbf{Dw}_2^{(i)}\ \ldots\ \mathbf{Dw}_S^{(i)}\right] \tag{25}$$

where S denotes the total number of training samples.

The learning convergence of the HCAQ-CMAC network is established via the convergence of the network memory contents as training approaches infinity. In this case, the sufficient and necessary condition for the HCAQ-CMAC learning process to convergence can be expressed in

$$\lim_{i\to\infty}\mathbf{Dw}_s^{(i)} = 0 \qquad \forall s \in \{1\cdots S\} \tag{26}$$

or

$$\lim_{i\to\infty}\mathbf{Dw}^{(i)} = [0] \tag{27}$$

where $[0]$ is the null matrix.

Substituting (24) into (23)

$$\mathbf{Dw}_s^{(i)} = \mathbf{E}_{s-1}\mathbf{Dw}_{s-1}^{(i)}. \tag{28}$$

The HCAQ-CMAC network is iteratively trained on a set of S training samples. When $s = 1$, from (28)

$$\mathbf{Dw}_1^{(i)} = \mathbf{E}_0\mathbf{Dw}_0^{(i)} \tag{29}$$

such that

$$\mathbf{Dw}_0^{(i)} = \mathbf{Dw}_S^{(i-1)} \tag{30}$$

and

$$\begin{aligned}&\mathbf{E}_0 = \mathbf{E}_S \\ \Rightarrow\ &\mathbf{C}_0 = \mathbf{C}_S \qquad \text{[from (24)]}.\end{aligned} \tag{31}$$

Following the results of (28)–(31), $\mathbf{Dw}^{(i)}$ [see (25)] can be reexpressed as

$$\begin{aligned}\mathbf{Dw}^{(i)} &= \left[\mathbf{Dw}_1^{(i)}\ \mathbf{Dw}_2^{(i)}\ \cdots\ \mathbf{Dw}_S^{(i)}\right] \\ &= \left[\underbrace{\mathbf{E}_S\mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}}\ \underbrace{\mathbf{E}_1\mathbf{Dw}_1^{(i)}}_{\mathbf{Dw}_2^{(i)}}\ \cdots\underbrace{\mathbf{E}_{S-1}\mathbf{Dw}_{S-1}^{(i)}}_{\mathbf{Dw}_S^{(i)}}\right] \\ &= \left[\mathbf{E}_S\underbrace{\mathbf{E}_{S-1}\mathbf{Dw}_{S-1}^{(i-1)}}_{\mathbf{Dw}_S^{(i-1)}}\ \mathbf{E}_1\underbrace{\mathbf{E}_S\mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}}\right. \\ &\qquad \left.\cdots\mathbf{E}_{S-1}\underbrace{\mathbf{E}_{S-2}\mathbf{Dw}_{S-2}^{(i)}}_{\mathbf{Dw}_{S-1}^{(i)}}\right].\end{aligned} \tag{32}$$

Decomposing the $\mathbf{Dw}$ terms on the right-hand side repeatedly results in

$$\begin{aligned}&\mathbf{Dw}^{(i)} \\ &= \left[(\mathbf{E}_S\mathbf{E}_{S-1}\cdots\mathbf{E}_1)\mathbf{Dw}_1^{(i-1)}\ (\mathbf{E}_1\mathbf{E}_S\cdots\mathbf{E}_2)\mathbf{Dw}_2^{(i-1)}\ \cdots\right. \\ &\qquad \left.(\mathbf{E}_{S-1}\mathbf{E}_{S-2}\cdots\mathbf{E}_S)\mathbf{Dw}_S^{(i-1)}\right].\end{aligned} \tag{33}$$

Following (33):

$$\mathbf{G}_s \equiv \mathbf{E}_{s-1}\mathbf{E}_{s-2}\cdots\mathbf{E}_1\mathbf{E}_S\mathbf{E}_{S-1}\ldots\mathbf{E}_s, \qquad s \in \{1\cdots S\}. \tag{34}$$

Therefore, (33) can be reexpressed as

$$\begin{aligned}&\mathbf{Dw}^{(i)} \\ &= \left[\mathbf{Dw}_1^{(i)}\ \mathbf{Dw}_2^{(i)}\ \cdots\ \mathbf{Dw}_S^{(i)}\right] \\ &= \left[\underbrace{(\mathbf{E}_S\mathbf{E}_{S-1}\cdots\mathbf{E}_1)}_{\mathbf{G}_1}\mathbf{Dw}_1^{(i-1)}\ \underbrace{(\mathbf{E}_1\mathbf{E}_S\cdots\mathbf{E}_2)}_{\mathbf{G}_2}\mathbf{Dw}_2^{(i-1)}\right. \\ &\qquad \left.\cdots\underbrace{(\mathbf{E}_{S-1}\mathbf{E}_{S-2}\cdots\mathbf{E}_S)}_{\mathbf{G}_S}\mathbf{Dw}_S^{(i-1)}\right] \\ &= \left[\mathbf{G}_1\mathbf{Dw}_1^{(i-1)}\ \mathbf{G}_2\mathbf{Dw}_2^{(i-1)}\ \cdots\ \mathbf{G}_S\mathbf{Dw}_S^{(i-1)}\right].\end{aligned} \tag{35}$$

It can be observed that

$$\mathbf{Dw}_s^{(i)} = \mathbf{G}_s\mathbf{Dw}_s^{(i-1)}, \qquad s \in \{1\cdots S\}. \tag{36}$$

Consequently, it follows that

$$\begin{aligned}\mathbf{Dw}^{(i)} &= \left[\mathbf{Dw}_1^{(i)}\ \mathbf{Dw}_2^{(i)}\ \cdots\ \mathbf{Dw}_S^{(i)}\right]\\ &= \left[\mathbf{G}_1\mathbf{Dw}_1^{(i-1)}\ \mathbf{G}_2\mathbf{Dw}_2^{(i-1)}\ \cdots\ \mathbf{G}_S\mathbf{Dw}_S^{(i-1)}\right]\\ &= \Bigg[\mathbf{G}_1\underbrace{\mathbf{G}_1\mathbf{Dw}_1^{(i-2)}}_{\mathbf{Dw}_1^{(i-1)}}\ \mathbf{G}_2\underbrace{\mathbf{G}_2\mathbf{Dw}_2^{(i-2)}}_{\mathbf{Dw}_2^{(i-1)}}\\ &\qquad \cdots\mathbf{G}_S\underbrace{\mathbf{G}_S\mathbf{Dw}_S^{(i-2)}}_{\mathbf{Dw}_S^{(i-1)}}\Bigg]\\ &= \left[(\mathbf{G}_1)^2\mathbf{Dw}_1^{(i-2)}\ (\mathbf{G}_2)^2\mathbf{Dw}_2^{(i-2)}\right.\\ &\qquad \left.\cdots(\mathbf{G}_S)^2\mathbf{Dw}_S^{(i-2)}\right]. \end{aligned} \tag{37}$$

Further repeated decomposition of the $\mathbf{Dw}$ terms on the right-hand side results in the following:

$$\mathbf{Dw}^{(i)} = \left[(\mathbf{G}_1)^i\mathbf{Dw}_1^{(0)}\ (\mathbf{G}_2)^i\mathbf{Dw}_2^{(0)}\ \cdots\ (\mathbf{G}_S)^i\mathbf{Dw}_S^{(0)}\right]. \tag{38}$$

With respect to (38), the memory difference matrix $\mathbf{Dw}^{(i)}$ must approach a null matrix as training tends to infinity (i.e., $i \to \infty$) in order to establish the learning convergence of the proposed HCAQ-CMAC network. Hence, the HCAQ-CMAC learning process converges if and only if (39) holds

$$\lim_{i\to\infty}(\mathbf{G}_s)^i\mathbf{Dw}_s^{(0)} = 0 \qquad \forall s \in \{1\cdots S\}. \tag{39}$$

By definition, the difference vector $\mathbf{Dw}_s^{(0)}$ can be expressed as

$$\begin{aligned}\mathbf{Dw}_s^{(0)} &= \mathbf{W}_s^{(1)} - \mathbf{W}_s^{(0)}\\ &= \underbrace{\mathbf{W}_{s-1}^{(1)} + \Delta\mathbf{W}_{s-1}^{(1)}}_{\mathbf{W}_s^{(1)}} - \mathbf{W}_s^{(0)}\\ &= \underbrace{\mathbf{W}_{s-2}^{(1)} + \Delta\mathbf{W}_{s-2}^{(1)}}_{\mathbf{W}_{s-1}^{(1)}} + \Delta\mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)}. \end{aligned} \tag{40}$$

Decomposing the $\mathbf{W}_s^{(i)}$ terms on the right-hand side repeatedly produces

$$\begin{aligned}\mathbf{Dw}_s^{(0)} &= \mathbf{W}_1^{(1)} + \Delta\mathbf{W}_1^{(1)} + \Delta\mathbf{W}_2^{(1)}\\ &\quad + \cdots + \Delta\mathbf{W}_{s-2}^{(1)} + \Delta\mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)}\\ &= \underbrace{\mathbf{W}_S^{(0)} + \Delta\mathbf{W}_S^{(0)}}_{\mathbf{W}_1^{(1)}} + \Delta\mathbf{W}_1^{(1)}\\ &\quad + \cdots + \Delta\mathbf{W}_{s-2}^{(1)} + \Delta\mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)}\\ &= \underbrace{\mathbf{W}_s^{(0)} + \Delta\mathbf{W}_s^{(0)}}_{\mathbf{W}_{s+1}^{(0)}} + \Delta\mathbf{W}_{s+1}^{(0)}\\ &\quad + \cdots + \Delta\mathbf{W}_S^{(0)} + \Delta\mathbf{W}_1^{(1)} + \cdots + \Delta\mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)}\\ &= \Delta\mathbf{W}_s^{(0)} + \Delta\mathbf{W}_{s+1}^{(0)} + \cdots + \Delta\mathbf{W}_S^{(0)}\\ &\quad + \Delta\mathbf{W}_1^{(1)} + \cdots + \Delta\mathbf{W}_{s-1}^{(1)}. \end{aligned} \tag{41}$$

From (22), the HCAQ-CMAC memory update due to the sth input training sample at the ith iteration ($\Delta\mathbf{W}_s^{(i)}$) is computed via

$$\Delta\mathbf{W}_s^{(i)} = \alpha\,\underbrace{\mathbf{C}_s\overbrace{\left\{\hat{Y}_s - \mathbf{C}_s^T\mathbf{W}_s^{(i)}\right\}}^{\text{learning error}}}_{\text{local error}} \tag{42}$$

where $\{\hat{Y}_s - \mathbf{C}_s^T\mathbf{W}_s^{(i)}\}$ is a scalar value and it is the learning (training) error of HCAQ-CMAC for the sth input training vector at the ith iteration. If

$$u_s^{(i)} = \left(\hat{Y}_s - \mathbf{C}_s^T\mathbf{W}_s^{(i)}\right) \tag{43}$$

then (42) can be reexpressed as

$$\Delta\mathbf{W}_s^{(i)} = \alpha\mathbf{C}_s u_s^{(i)}. \tag{44}$$

From (38), (41), and (44)

$$\begin{aligned}(\mathbf{G}_s)^i\mathbf{Dw}_s^{(0)} &= (\mathbf{G}_s)^i\left\{\Delta\mathbf{W}_s^{(0)} + \Delta\mathbf{W}_{s+1}^{(0)} + \cdots + \Delta\mathbf{W}_S^{(0)}\right.\\ &\qquad \left. + \Delta\mathbf{W}_1^{(1)} + \cdots + \Delta\mathbf{W}_{s-1}^{(1)}\right\}\\ &= (\mathbf{G}_s)^i\left\{\alpha\mathbf{C}_s u_s^{(0)} + \alpha\mathbf{C}_{s+1}u_{s+1}^{(0)} + \cdots\right.\\ &\qquad \left. + \alpha\mathbf{C}_{s-1}u_{s-1}^{(1)}\right\}\\ &= \alpha(\mathbf{G}_s)^i\left\{\mathbf{C}_s u_s^{(0)} + \mathbf{C}_{s+1}u_{s+1}^{(0)} + \cdots\right.\\ &\qquad \left. + \mathbf{C}_{s-1}u_{s-1}^{(1)}\right\}, \qquad s \in \{1\cdots S\}. \end{aligned} \tag{45}$$

Therefore, if $\lim_{i\to\infty}(\mathbf{G}_s)^i\mathbf{C}_a = [0]$ for all $a \in \{1\cdots S\}, (\mathbf{G}_s)^i\mathbf{Dw}_s^{(0)}$ in (45) evaluates as null. From (38), the matrix $\mathbf{Dw}^{(i)} = [0]$ as $i \to \infty$ follows. Consequently, the learning process of the proposed HCAQ-CMAC network converges.

B. Learning Convergence of the HCAQ-CMAC Network

Theorem 1: The training process of the HCAQ-CMAC network converges if and only if the learning constant α is such that $0 < \alpha < 2$.

Proof: It can be shown that for all $a \in \{1\cdots S\}$, when $0 < \alpha < 2$, $\lim_{i\to\infty}(\mathbf{G}_s)^i\mathbf{C}_a = [0]$ (reefer to Appendix B for a detailed proof). Therefore, the training process of the HCAQ-CMAC network converges if and only if the learning constant α satisfies the condition $0 < \alpha < 2$. ■

VI. Experimental Results and Analysis

This section presents the experiments that have been conducted to evaluate the performance of the proposed HCAQ-CMAC network. The experiments are performed on the following two real-life applications: 1) automatic control of car maneuver and 2) modeling of the human glucose metabolic process. The performances of the HCAQ-CMAC network are benchmarked against 1) the basic CMAC network to demonstrate the memory efficiency achievable by the proposed HCAQ-CMAC network, 2) Moody's multiresolution CMAC

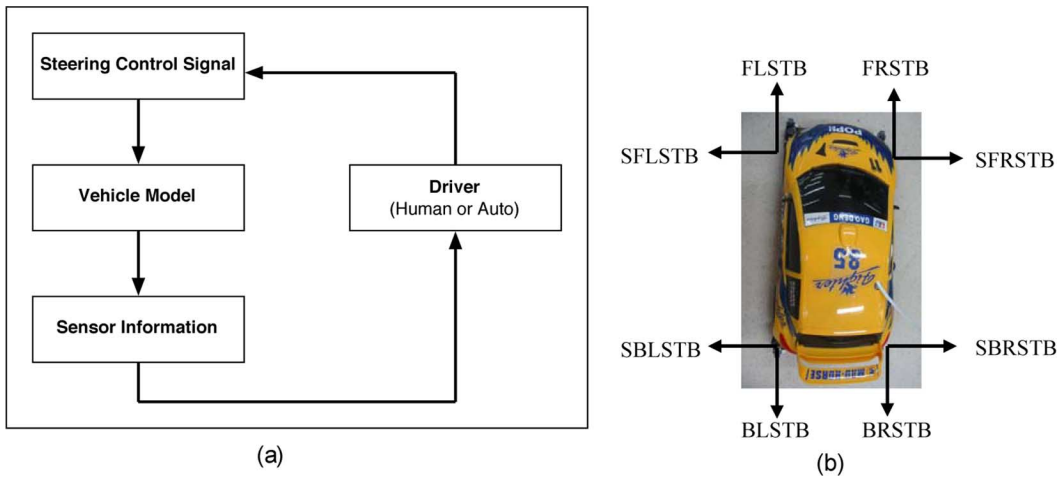


Fig. 9. Simulation environment. (a) Vehicle control sequence. (b) Car sensor placements.

network [25] to assess the generalization and learning ability of the proposed network, and 3) Menozzi's tree-based multiresolution CMAC [24] to assess both the modeling performance and memory efficiency achievable by HCAQ-CMAC.

A. Automatic Control of Car Maneuver

The *intelligent vehicle project* is part of an ongoing research effort to develop an intelligent transportation system (ITS) at the Centre for Computational Intelligence (C2i), Nanyang Technological University, Singapore[55]. The objective of the project is to realize brain-inspired intelligent-based technologies required for the automation of control, routing, and navigation of land vehicles. In this paper, the proposed HCAQ-CMAC network is employed for the construction of an autopilot system for car maneuver. Albeit its complexity, driving a vehicle is a motor task that humans are able to perform relatively well. It has been well established that the learning of motor skills is mediated by the human *procedural memory system* [56], which consist of the cerebellum and the striatum (part of basal ganglia formation). The human procedural memory system is a facet of the brain's information processing capacity specifically for the acquisition of skilled behaviors and habits. Vehicle driving comprises of finely tuned sets of sensory feedback to control action mappings that are accumulated through experiences and repeated practices. Although humans are quite adept at mastering complex skills, it is difficult to formalize these behaviors into mathematical algorithms. In such cases, the construction of a computational model that is able to emulate the functionality of the human brain is required [57]. This subsequently motivates the use of HCAQ-CMAC to model and emulate the human driving expertise.

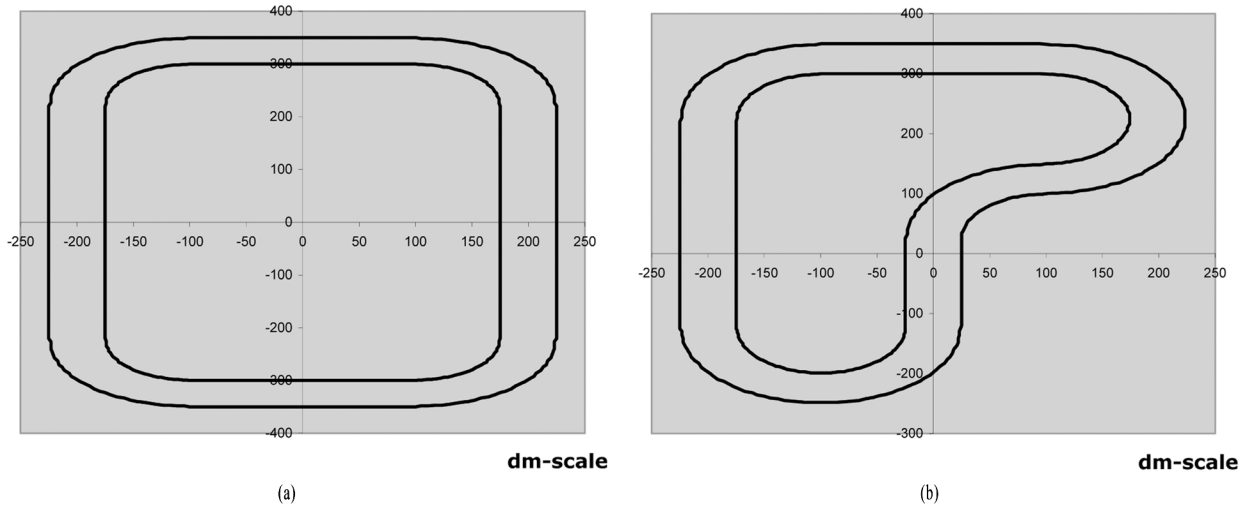
In this experiment, a driving simulator (as developed in [58]) is employed to capture the behavioral response of the human driver. The simulator consists of a 3-D virtual driving environment that integrates a detailed model of the vehicle dynamics and engine characteristics together with the environmental parameters such as road profiles. The approach to the experiment is to capture and record the driving data of a human driver, which consists of a set of distance-sensor feedback information and the corresponding steering control performed by the human driver as he maneuvers the simulated car around a specified track. The feedback signals provide information such as the distance of the vehicle from the road boundaries. The simulator allows for vehicle control via steering adjustment. An overview of the driving control sequence is given in Fig. 9(a). These feedback-control records are then used to train the HCAQ-CMAC-based autopilot system.

The objective of the autopilot system is to control the vehicle to follow a particular lane in a multilane circuit track. The simulated vehicle model is equipped with eight directional sensors as shown in Fig. 9(b). The semantics of the sensor readings are tabulated as Table I. For the autopilot system, only the front four sensors are utilized [i.e., SFLSTB, FLSTB, FRSTB, SFRSTB; Fig. 9(b)] as the inputs to the HCAQ-CMAC network. As an output, the network responds with the appropriate steering angle. Two tracks are used in this experiment. The HCAQ-CMAC autopilot system is first trained on track 1 [Fig. 10(a)], with the car traveling in both clockwise and counterclockwise direction. Subsequently, track 2 [Fig. 10(b)] is used for testing. The simulated track is 5 m wide.

The training data set (recorded from a human driver) contains 1018 samples. HCAQ-CMAC and the benchmarked systems were trained within 500 training epochs with a network learning constant of 0.1. Table II outlines the performances of the HCAQ-CMAC autopilot system as compared to the following: 1) the basic CMAC network, 2) Moody's multiresolution CMAC network, and 3) the tree-based multiresolution CMAC network. Each simulation test result was collected over 100 s of driving-time, with a maximum driving speed of 100 km/h. The driving performances of the various networks were measured by the average deviation of the controlled car from the center of the lane (ACD) and the average deviation of the car orientation from the desired orientation (AOD). Both the ACD and AOD measurements are subsequently normalized and reported as normalized ACD (NACD) and normalized AOD (NAOD). NACD denotes the ACD with respect to the half track width, which is the maximum leeway available before the car collides with the road boundaries. The AOD values are normalized with respect to π radians. A performance index (PI_1) is used to combine the NACD and NACD measures as described in

$$\mathrm{PI}_1 = (100 - \mathrm{NACD}) + (100 - \mathrm{NAOD}) \tag{46}$$

$$\widehat{\mathrm{PI}}_1 = \mathrm{PI}_1/200 \tag{47}$$



(a) (b)

Fig. 10. Driving tracks. (a) Training track (track 1). (b) Testing track (track 2).

TABLE I
SENSORS DEFINITION OF THE CAR SIMULATOR

Symbol	Explanation
FLSTB	Front Left Sensor to Barrier
FRSTB	Front Right Sensor to Barrier
SFLSTB	Side Front Left Sensor to Barrier
SFTSTB	Side Front Right Sensor to Barrier
SBLSTB	Side Back Left Sensor to Barrier
SBRSTB	Side Back Right Sensor to Barrier
BLSTB	Back Left Sensor to Barrier
BRSTB	Back Right Sensor to Barrier

where $\widehat{\text{PI}}_1$ is the normalized PI_1. Thus, a higher PI_1 value corresponds to a better network performance.

For the simulation, a learning constant of 0.1 and a neighborhood constant of 0.2 were empirically determined for the evaluated networks. The network size is also varied to determine the optimal performance for each of the networks. Based on the PI_1 values in Table II, the basic CMAC network achieved an optimal performance with a memory size of ten memory cells per dimension. For CMAC, a small network size results in a coarse partitioning of the input space, and hence, the network suffers from an *averaging effect*. A large CMAC network, however, fails to generalize from the available training data and thus performs poorly on the testing track. The multiresolution CMAC architectures, on the other hand, employ layers of overlapping CMAC networks with different resolutions to address the generalization-accuracy dilemma. Both Moody's multiresolution CMAC (MMR-CMAC) and the tree-based multiresolution CMAC (TMR-CMAC) architectures were benchmarked using two- and three-layers implementations. These overlay implementations improve the generalization ability of the finer resolution CMACs while simultaneously maintaining their output accuracy, but at the expense of higher memory requirements. Best PI_1 values were observed for a two-layer MMR-CMAC with memory size of five cells per dimension for the first layer and ten cells per dimension for the secondary layer, and a two-level TMR-CMAC employing a CMAC of size eight cells per dimension at the first level and 26 CMACs of size two per dimension at the second level.

From the results presented in Table II, one can observe that the proposed HCAQ-CMAC-based autopilot system consistently outperformed the other three CMAC architectures. An optimal HCAQ-CMAC network performance is obtained with a memory size of only five cells per dimension with a PI_1 value of 172. The hierarchical clustering technique of the HCAQ-CMAC network effectively allocates the available memory cells to the input regions with high utilization throughput. This ensures that more cells are allocated to important input regions that contain more information. Therefore, a small network size does not affect the accuracy and fine-tuning capability of the HCAQ-CMAC network. Moreover, with a smaller network size, fewer cells in HCAQ-CMAC are allocated to areas with little or no training data, thus improving the generalization ability of the network. The efficient memory allocation scheme of HCAQ-CMAC also reduces the required network training time. The best-performing HCAQ-CMAC network trains in the shortest time (i.e., 7562 ms) among all the benchmarked networks. Subsequently, the effective cell utilization rates of the various networks were computed and the results [denoted as the cell occupancy rate (COR)] are tabulated in Table III. The COR is defined as the proportion of the trained network cells to the total network size. From Table III, one can observe that the HCAQ-CMAC network achieves the highest COR value in comparison with the other three CMAC architectures. This clearly demonstrated the effectiveness of the proposed HCAQ-CMAC memory allocation scheme.

B. Modeling the Dynamics of the Human Glucose Metabolic Process

Diabetes is a chronic disease where the body is unable to properly and efficiently regulate the use and storage of glucose in the blood. This resulted in large perturbations of the plasma glucose level, leading to *hyperglycemia* (elevated glucose level) or *hypoglycemia* (depressed glucose level). Chronic hyperglycemia causes severe damage to the eyes, kidneys, nerves, heart, and blood vessels of the patients while severe

TABLE II
COMPARISON OF RESULTS FOR THE VARIOUS CMAC NETWORKS USING THE AUTOPILOT SYSTEM

Networks	Memory size (4 input dimensions)	Train Time [ms]	ACD [m]	NACD [%]	AOD [rads]	NAOD [%]	PI_1	$\widehat{PI}_1$
CMAC	7^4	8749	0.2694	10.78	0.7388	23.53	165.60	0.8280
	8^4	9031	0.3497	13.99	0.7272	23.16	162.85	0.8143
	9^4	collided						
	$\mathbf{10^4}$	**14453**	**0.2068**	**8.27**	**0.7594**	**24.18**	**167.55**	**0.8378**
	11^4	21687	0.3131	12.52	0.7407	23.59	163.89	0.8195
	12^4	87531	0.4071	16.28	0.7016	22.34	161.38	0.8069
HCAQ-CMAC	$\mathbf{5^4}$	**7562**	**0.1501**	**6.00**	**0.6908**	**22.00**	**172.00**	**0.8600**
	6^4	9109	0.1591	6.36	0.7145	22.75	170.89	0.8545
	7^4	14234	0.1557	6.23	0.7287	23.20	170.59	0.8530
	8^4	25796	0.2048	8.19	0.7316	23.30	168.51	0.8426
	9^4	31297	0.2079	8.32	0.7453	23.74	167.94	0.8397
	10^4	55202	0.2148	8.59	0.7283	23.19	168.22	0.8411
MMR-CMAC	$2^4 + 8^4$	19234	0.4054	16.22	0.7322	23.32	160.64	0.8032
	$4^4 + 8^4$	22813	0.4451	17.80	0.7250	23.09	159.11	0.7956
	$3^4 + 10^4$	27328	0.2085	8.34	0.7674	24.44	167.22	0.8361
	$\mathbf{5^4 + 10^4}$	**26204**	**0.2056**	**8.22**	**0.7497**	**23.88**	**167.90**	**0.8395**
	$2^4 + 4^4 + 8^4$	30718	0.4770	19.08	0.7071	22.52	158.40	0.7920
	$2^4 + 3^4 + 10^4$	36484	0.3253	13.01	0.7814	24.89	162.10	0.8105
	$2^4 + 5^4 + 10^4$	34687	0.3164	12.66	0.7689	24.49	162.85	0.8143
TMR-CMAC	$\mathbf{8^4 + (26 \times 2^4)}$	**24875**	**0.3552**	**14.21**	**0.7378**	**23.50**	**162.29**	**0.8115**
	$8^4 + (10 \times 2^4) + (16 \times 4^4)$	36328	0.4093	16.37	0.6944	22.11	161.52	0.8076
	$8^4 + (6 \times 4^4) + (10 \times 8^4)$	52859	0.4093	16.37	0.6996	22.28	161.35	0.8068

**CMAC* denotes the basic CMAC network, *HCAQ-CMAC* denotes the proposed HCAQ-CMAC network, *MMR-CMAC* denotes Moody's Multi-Resolution CMAC network, while *TMR-CMAC* is the Tree-based Multi-Resolution CMAC. *ACD* is the average deviation from the center of the lane, *NACD* is the normalized ACD, *AOD* denotes the average deviation of the orientation, *NAOD* is the normalized AOD, PI_1 denotes the Performance Index, and $\widehat{PI}_1$ is the normalized PI_1.

hypoglycemia can deprive the body of energy and causes the patient to lose consciousness, which can eventually become life threatening. Currently, the treatment of diabetes is based on a two-pronged approach: strict dietary control and insulin medication.

The key component to a successful management of diabetes is essentially to develop the ability to maintain a long-term near-*normoglycaemia* state of the patient. With respect to this objective, the therapeutic effect of discrete insulin injections is not ideal as the regulation of insulin is an open-looped process. Continuous insulin infusion through an insulin pump, on the other hand, is a more viable approach due to its controllable infusion rate [59]. Such insulin pumps are algorithmic-driven, with an avalanche of techniques proposed, investigated and reported in the literature over the years [60], [61]. Generally, all such proposed methods required some forms of *accurate* modeling of the glucose metabolic process of the diabetic patient before a suitable control regime can be devised.

In recent years, emerging evidences have suggested that glucose metabolism throughout the body is coordinated by the brain through the use of insulin [62]. This is reinforced by the fact that glucokinase, the established glucose sensor of the pancreatic β-cells, is observed to be also present in the central nervous system (CNS) [63]. Precise experimentations have subsequently demonstrated that insulin, via acting on the hypothalamus (a subcortical brain structure central to the autonomic control of the human endocrine system), exerts a high level of supervisory control on glucose production by the liver [64]. This observation contemplates that insulin can mediate the human glucose metabolic process through an unknown signaling pathway via the CNS [65], [66]. This notion subsequently motivates the use of the HCAQ-CMAC network, which is a brain-inspired computational model of the human cerebellum, for the dynamic modeling of the human blood glucose cycle.

The first step into constructing a model of the human glucose metabolic process is to determine the patient profile to be modeled. Due to the lack of real-life patient data and the logistical difficulties and ethical issues involving the collection of such data, a well-known web-based simulator known as *GlucoSim* [67] from the Illinois Institute of Technology (IIT, Chicago, IL), is employed to simulate a person subject to generate the blood glucose data that is needed for the construction of the glucose metabolism model. A person profile for the simulated healthy subject is created as shown in Table IV.

The simulated healthy person, code-named subject A, is a typical middle-aged Asian male. His body mass index (BMI) is 23.0, which is within the recommended range for Asian. Based on the person profile of subject A, his recommended daily allowance (RDA) of carbohydrate intake from meals is obtained from the website of the Health Promotion Board of Singapore [68]. According to his sex, age, weight, and lifestyle, the recommended daily carbohydrate intake for subject A is approximately 346.9 g.

TABLE III
COMPARISON OF CORs FOR THE VARIOUS CMAC NETWORKS

Networks	Memory Size	Total Cells	Trained Cells	Cell Occupancy Rate
CMAC	7^4	2401	92	3.83%
	8^4	4096	102	2.49%
	9^4	6561	collided	
	$\mathbf{10^4}$	**10000**	**152**	**1.52%**
	11^4	14641	176	1.2%
	12^4	20736	186	0.9%
HCAQ-CMAC	$\mathbf{5^4}$	**625**	**73**	**11.68%**
	6^4	1296	89	6.87%
	7^4	2401	127	5.29%
	8^4	4096	161	3.93%
	9^4	6561	172	2.62%
	10^4	10000	214	2.14%
MMR-CMAC	$2^4 + 8^4$	4112	106	2.58%
	$4^4 + 8^4$	4352	130	2.99%
	$3^4 + 10^4$	10081	166	1.65%
	$\mathbf{5^4 + 10^4}$	**10625**	**197**	**1.85%**
	$2^4 + 4^4 + 8^4$	4368	134	3.07%
	$2^4 + 3^4 + 10^4$	10097	169	1.67%
	$2^4 + 5^4 + 10^4$	10641	201	1.88%
TMR-CMAC	$\mathbf{8^4 + (26 \times 2^4)}$	**4512**	**182**	**4.03%**
	$8^4 + (10 \times 2^4) + (16 \times 4^4)$	8352	220	2.63%
	$8^4 + (6 \times 4^4) + (10 \times 8^4)$	46592	220	0.47%

**CMAC* denotes the basic CMAC network, *HCAQ-CMAC* denotes the proposed HCAQ-CMAC network, *MMR-CMAC* denotes Moody's Multi-Resolution CMAC network, while *TMR-CMAC* is the Tree-based Multi-Resolution CMAC.

TABLE IV
PROFILE OF THE SIMULATED HEALTHY PERSON (SUBJECT A)

Attribute Name	Attribute Value
Sex	Male
Age	40 years old
Race	Asian
Weight	67 kg (147.71 lbs)
Height	1.70 m (5 ft 7 in)
BMI	23 (Recommended for Asian)
Lifestyle	Typical office worker with moderate physical activities such as walking briskly, leisure cycling and swimming.

Fig. 11 illustrates a sample output from GlucoSim for subject A. This output consists of six elements: blood glucose, blood insulin, intestinal glucose absorption rate, stomach glucose, total glucose uptake rate, and liver glucose production rate of subject A, respectively, over a simulated time period of 24 h. The peaks in the stomach glucose subplot of Fig. 11 coincide with the timings of the assumed four daily meals (i.e., breakfast, lunch, afternoon snack, and dinner) while those peaks in the intestinal glucose absorption rate subplot reflect a delay effect (response) of food intake on the blood glucose level of subject A. The subplots of blood glucose and blood insulin illustrate the insulin–glucose regulatory mechanism in a healthy person such as subject A and depict the dynamics of the metabolic process when subjected to disturbances such as food intakes.

Since the human glucose metabolic process depends on its own current (and internal) states as well as the exogenous food intakes, it is hypothesized that the blood glucose level at any given time is a nonlinear function of prior food intakes and the historical traces of the insulin and blood glucose levels. To properly account for the effects of prior food ingestions to the fluctuation of the blood glucose level, a historical window of 6 h is adopted to trace the carbohydrate content of the meals taken. A soft-windowing strategy is employed to temporally partition the 6-h historical window into three conceptual segments, namely, *recent* (i.e., previous 1 h), *intermediate past* (i.e., previous 1–3 h), and *long ago* (i.e., previous 3–6 h). Based on these windows, three normalized weighting functions are introduced to compute the carbohydrate content of the meal(s) (with respect to current time) taken recently, in the intermediate past or long ago. Thus, inclusive of the measured blood glucose and insulin levels, there is a total of five inputs to the modeling task. Fig. 12 depicts the weighting function for each of the respective segmented windows.

Based on the formulated hypothesis and the preprocessed glucose data generated from GlucoSim, a total of 100 days of glucose metabolic data for subject A was collected. The carbohydrate content and the timings of the daily meals were varied on a daily basis during the data collection phase. This ensures that HCAQ-CMAC and the benchmarked networks are not trained on a cyclical data set, but are employed to model the inherent relationships between food intakes and the glucose metabolic process of a healthy person. The collected data set is partitioned into two nonoverlapping groups: 20 days of data for training and the remaining 80 days for testing and evaluation of the networks.

Simulations to model the dynamics of the blood glucose level of subject A using the HCAQ-CMAC network were performed and the results were benchmarked against those of the basic CMAC network, the MMR-CMAC network as well

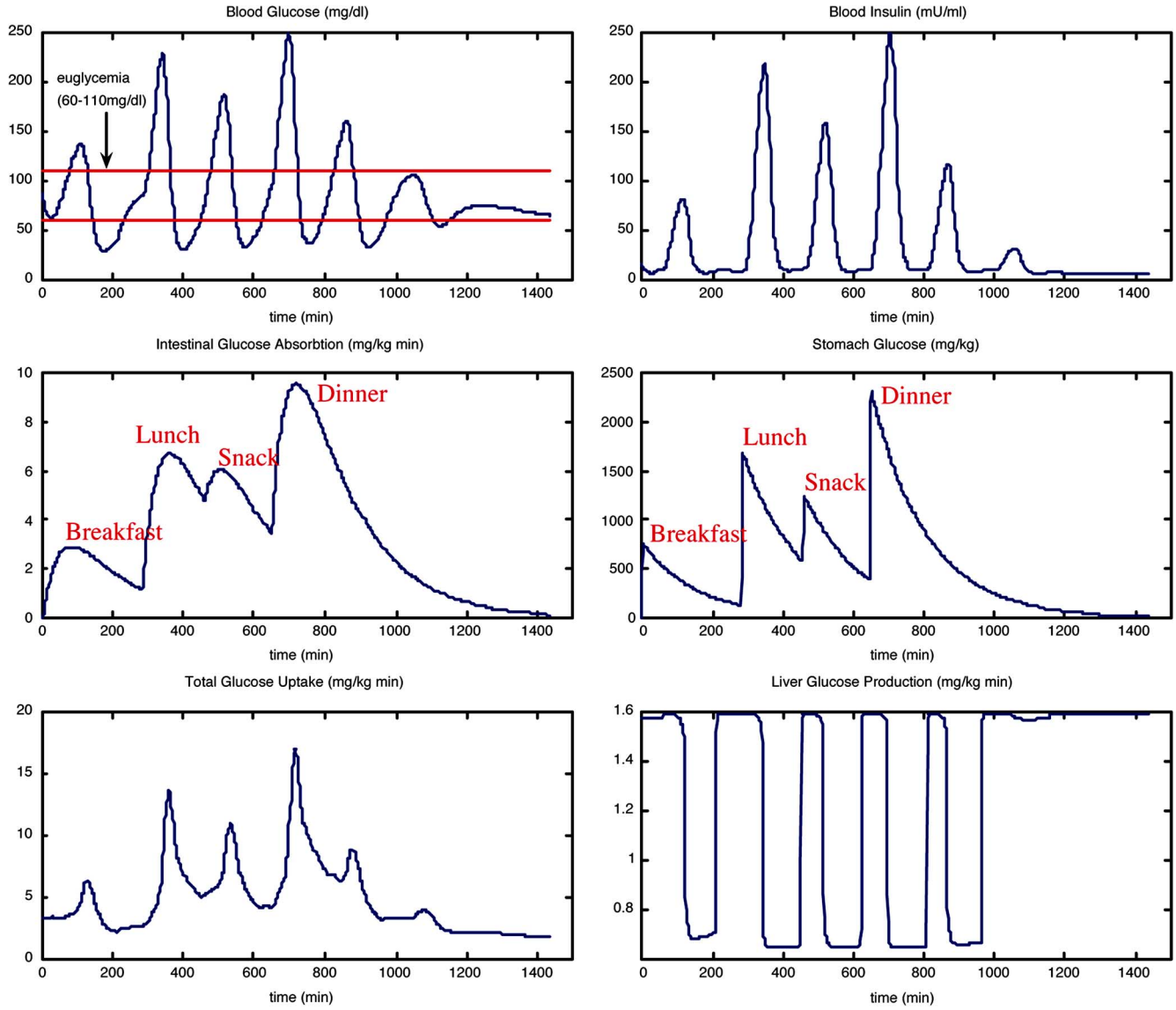


Fig. 11. Sample glucose metabolism data output from the GlucoSim simulator.

as the TMR-CMAC network. For this application, a neighborhood constant (N) of 0.1 and a learning rate (α) of 0.1 are empirically determined. Table V details the recall (training) and generalization (testing) performances of the various networks with different network sizes. Two performance indicators are employed to quantify the modeling quality of the networks: the root-mean-squared error (RMSE) and the *Pearson correlation coefficient* between the actual and the computed blood glucose level. The RMSE and Pearson correlation measures were subsequently employed to compute the PI_2 described by

$$\text{PI}_2 = \frac{\text{Pearcorr}}{(1+\text{RMSE})} \times 100. \tag{48}$$

Therefore, a higher PI_2 value reflects a better network modeling performance.

Due to the characteristics of the training data, the performances of the networks vary with their respective network sizes. From the PI_2 values in Table V, one can observe that for the basic CMAC network, an optimal performance (for generalization) is achieved with a memory size of four cells per dimension. For the basic CMAC network, a memory size of three cells per dimension is too small to extract the important details from the training data, while memory sizes with six to ten cells per dimension have too high a resolution to effectively capture the general trends. The optimal configuration for the MMR-CMAC in this experiment was found to be three layers of overlapping CMACs with network sizes of two, four, and eight cells per dimension, respectively. The experimental results in Table V indicated that the MMR-CMAC network required a large number of overlapping CMACs with different resolutions to obtain a good modeling performance. This can be observed by comparing the PI_2 value of the two-layered MMR-CMAC of size two and eight cells per dimension to that of the three-layered MMR-CMAC of size two, four, and eight cells per dimension, respectively. Although these two MMR-CMAC networks have the same base and top layer resolutions, a significant improvement in the PI_2 value was achieved by the three-layered

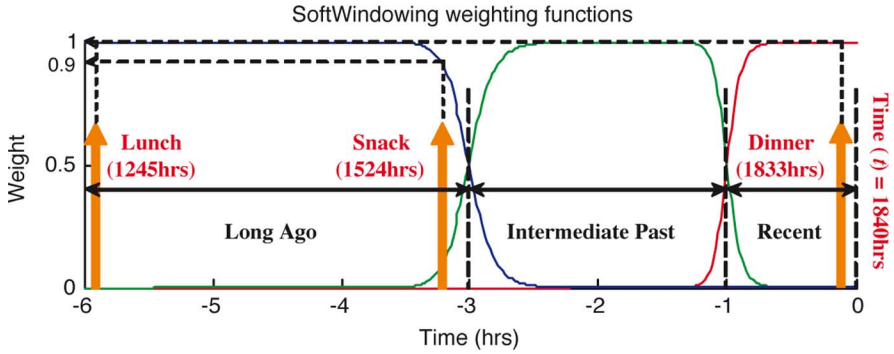


Fig. 12. Soft-windowing weighting functions to compute the carbohydrate content of meal(s) in the segmented windows of the 6-h food history. (In the figure, the current time is 1840 h and there are three instances of food intakes in the previous 6 h; that is, lunch at 1245 h, afternoon snack at 1524 h, and dinner at 1833 h, respectively.)

TABLE V
COMPARISON OF RESULTS FOR THE VARIOUS CMAC NETWORKS ON THE MODELING OF GLUCOSE METABOLISM PROCESS

Networks	Memory size	Recall			Generalization		
		RMSE [mg/ml]	Pearcorr	PI_2	RMSE [mg/ml]	Pearcorr	PI_2
CMAC	3^5	20.6009	0.8717	4.04	20.2713	0.8944	4.18
	$\mathbf{4^5}$	**16.3755**	**0.9210**	**5.30**	**17.8086**	**0.9190**	**4.89**
	6^5	11.1891	0.9639	7.91	22.9020	0.8608	3.60
	8^5	8.1866	0.9808	10.66	26.0175	0.8200	3.04
	10^5	6.1730	0.9892	13.79	26.2614	0.8156	2.99
HCAQ-CMAC	6^5	12.8754	0.9519	6.86	12.3468	0.9661	7.24
	7^5	11.2529	0.9635	7.86	11.0548	0.9724	8.07
	8^5	10.0242	0.9711	8.81	10.5101	0.9748	8.47
	9^5	10.0217	0.9712	8.81	10.4405	0.9752	8.52
	$\mathbf{10^5}$	**9.9606**	**0.9715**	**8.86**	**10.4312**	**0.9748**	**8.53**
MMR-CMAC	2^5+4^5	16.3755	0.9210	5.30	14.7504	0.9489	6.02
	2^5+6^5	11.1891	0.9639	7.91	12.0470	0.9630	7.38
	2^5+8^5	8.1866	0.9808	10.68	11.3426	0.9668	7.83
	$\mathbf{2^5+4^5+8^5}$	**8.1866**	**0.9808**	**10.68**	**10.3823**	**0.9724**	**8.54**
TMR-CMAC	$(3\times 2^5)+(8\times 4^5)$	8.2073	0.9807	10.65	34.1890	0.7014	1.99
	$\mathbf{3^5+(21\times 2^5)+(3\times 4^5)}$	**10.2723**	**0.9697**	**8.60**	**30.1240**	**0.7767**	**2.50**
	$4^5+(49\times 2^5)$	8.2493	0.9806	10.60	30.7527	0.7631	2.40
	$6^5+(45\times 2^5)$	7.7855	0.9827	11.19	32.2617	0.7398	2.22

**CMAC* denotes the basic CMAC network, *HCAQ-CMAC* denotes the proposed HCAQ-CMAC network, *MMR-CMAC* denotes Moody's Multi-Resolution CMAC network, while *TMR-CMAC* is the Tree-based Multi-Resolution CMAC. *Recall* refers to in-sample testing, *Generalization* is out-of-sample testing, PI_2 denotes the Performance Index.

MMR-CMAC via the addition of the middle-layer CMAC. On the other hand, even though the TMR-CMAC network also employed layers of CMAC networks of different resolutions, its performances were found to be relatively poor in comparison to the other benchmarked networks. This may be due to the overlay mechanism of the TMR-CMAC, where finer resolution CMACs are selectively allocated to the input subregions with large output errors. The resolutions of these CMACs are increased as required and only the finest resolution layers are kept.

On the other hand, an optimal PI_2 value (generalization) was achieved by the HCAQ-CMAC network with a memory size of ten cells per dimension. This HCAQ-CMAC network managed to produce a rather good fit to the actual blood glucose profile as indicated by a high correlation value of 97.48% and a relatively low RMSE of 10.4312 mg/mL of blood glucose concentration. There are only slight improvements in the HCAQ-CMAC performances as the network size is increased from six to ten cells per dimension. However, Table V clearly shows that the generalization capability of the proposed HCAQ-CMAC network surpasses that of the basic CMAC network for all evaluated memory sizes. Therefore, depending on the memory consideration and the desired modeling accuracy, an HCAQ-CMAC network with memory size of six to ten cells per dimension can be chosen for the modeling task. From Table V, one can observe that the three-layered MMR-CMAC achieves comparable performance to the HCAQ-CMAC network, while employing less memory cells. However, the overlay structure of the MMR-CMAC requires each layer of the network to be trained individually, thereby increasing the training time required. Furthermore, this overlay structure also makes output interpretation as well as hardware implementation awkward and more difficult as compared to the single-layered HCAQ-CMAC structure. In

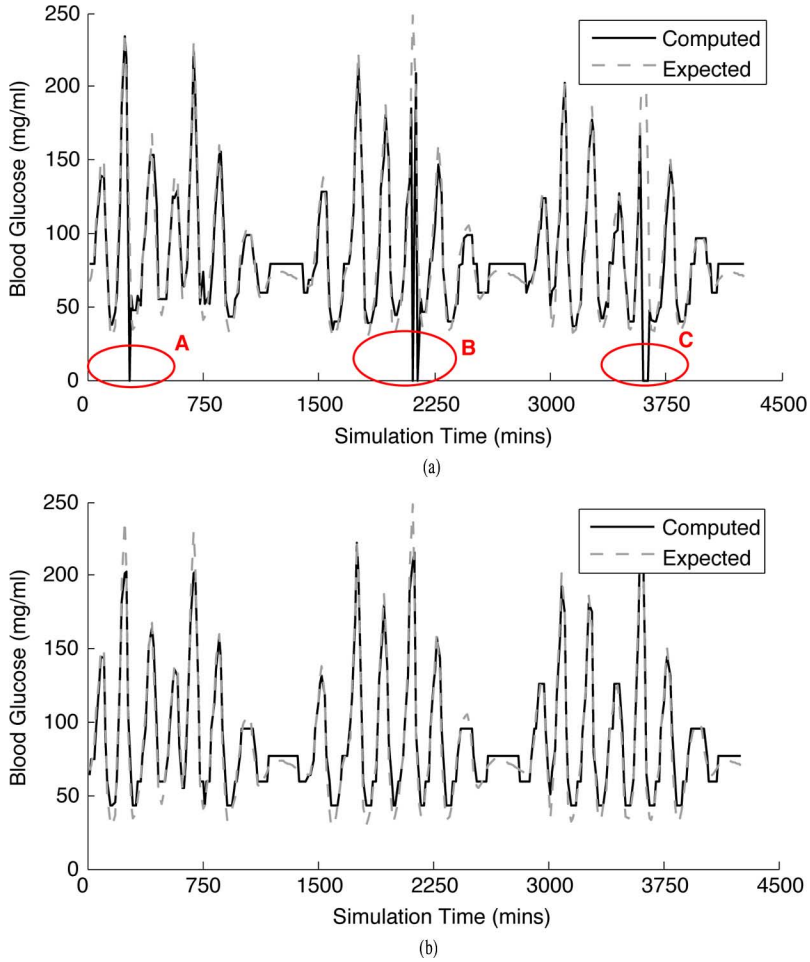


Fig. 13. Modeling results of the CMAC and HCAQ-CMAC network for the glucose metabolic process of subject A. (a) Generalization performance of CMAC network (size = ten cells per dimension). (b) Generalization performance of HCAQ-CMAC network (size = ten cells per dimension).

addition, the proposed HCAQ-CMAC network achieved comparable recall and generalization performances, which demonstrates that the network is able to efficiently extract the inherent relationships from the training data.

To further analyze the performance of the HCAQ-CMAC network, a three-days modeling result of the CMAC and HCAQ-CMAC networks (each of size ten cells per dimension) are depicted in Fig. 13. Fig. 13 clearly demonstrated the generalization and modeling accuracy of the HCAQ-CMAC network. Due to the nature of the training data, the static uniform quantization of the basic CMAC network resulted in an abundance of untrained network cells as evidenced in Fig. 13(a) (the effects of the untrained network cells are highlighted as $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, respectively). The HCAQ-CMAC network, on the other hand, is able to selectively allocate the available memory space according to the information distribution of the training data, and thus, significantly reduces the number of untrained network cells to facilitate a consistent modeling performance as evident in Fig. 13(b).

VII. Conclusion

This paper presents a novel brain-inspired nonuniformly quantized CMAC architecture named the HCAQ-CMAC network. Inspired by the physiology of the human cerebellum, as well as the neurobiological mechanisms of the neuronal selection process underlying human brain development, the HCAQ-CMAC network employs an information-driven memory allocation scheme.

The proposed HCAQ-CMAC network extends from the basic CMAC model by employing a hierarchical clustering technique to selectively allocate more memory cells to the input regions that contain more training information as reflected by the variations in the target output. This translates to a finer output resolution in the critical regions of the input space. The mathematical description of the formation of the memory structure and the subsequent learning process, together with the theoretical proof of HCAQ-CMAC learning stability, are presented in this paper.

The performance of the HCAQ-CMAC network was subsequently evaluated in two real-life applications, namely, the automatic control of car maneuver and the modeling of human blood glucose dynamics. Simulation results have sufficiently demonstrated the effectiveness of the proposed network architecture in capturing the complex I/O relationships for both applications. In particular, significant improvements in the generalization as well as the accuracy of the network output were achieved by the HCAQ-CMAC network. Moreover, HCAQ-CMAC also resulted in more efficient memory utilization than the benchmarked systems of CMAC, MMR-CMAC, and TMR-CMAC

TABLE VI
CORRESPONDENCE BETWEEN THE NEUROPHYSIOLOGICAL ASPECTS OF THE HUMAN CEREBELLUM AND THE FUNCTIONALITIES OF THE PROPOSED HCAQ-CMAC NETWORK

Characteristics	The Cerebellum	The HCAQ-CMAC Network
Physical connectivity	Purkinje cells are the main computational units of the human cerebellum and the parallel fiber inputs run perpendicularly to the flat fan-like dendritic aborization of the Purkinje cells.	Memory cells of the HCAQ-CMAC network are analogous to the Purkinje cells in the human cerebellum and the grid-like organization of these memory cells is inspired by the anatomy of the biological interconnections of the Purkinje cells and the parallel fibers.
Output computation	The output of the cerebellum originates from the deep cerebellar nuclei, which combine the outputs of the activated Purkinje cells.	The output of the HCAQ-CMAC network is a linear combination of the weighted contents of the activated memory cells.
Functionality	The cerebellum performs associative mapping from the input sensory afferent and cerebral efferent signals to the output of the brain construct	The HCAQ-CMAC network performs associative mapping from the input vector of the network to the output response.
Synaptic organization	Research into the neurophysiological aspects of the human brain have established that the precise wirings in an adult human brain are laid out as the results of competition-based neuronal selection process.	The HCAQ-CMAC network employs hierarchical clustering technique to perform selection of memory allocation in which more memory cells are allocated to input segments that contain more information from the training data.
Learning principle	The human cerebellum adopts an error-correction-driven supervised learning paradigm.	The HCAQ-CMAC network adopts the modified Widrow-Hoff training algorithm, which is essentially an error-correction based supervised learning scheme.
Learning convergence	Learning stability is vital since the cerebellum is responsible for smooth and precise coordination of the motor movements.	Learning stability (convergence) in the HCAQ-CMAC network has been established (Refer to Section V).

as reflected by the considerably higher CORs observed in the car-driving experiment.

However, even though HCAQ-CMAC reports relatively higher CORs, memory efficiency remains at less than 12%. This is mainly due to the fact that the quantization process of the HCAQ-CMAC is separately performed in each input dimension. This is to reduce the computational complexity arising from the introduction of nonuniform quantization to the CMAC network, and work reasonably well for low-dimensional problems but may cause large memory wastage for higher dimensional problems. Research efforts are currently directed at addressing this limitation. Currently, the memory sizes employed in the applications have been empirically determined. This is because depending on the data characteristics, different applications require different memory sizes to achieve an optimal performance. The hierarchical clustering technique employed in the HCAQ-CMAC network does not possess the ability to automatically determine the optimal number of quantization clusters. Instead, this paper concentrates on achieving maximum performance with the available (predefined) number of memory cells. A future enhancement to HCAQ-CMAC is to extend the architecture to support online learning. Currently, since the result of the hierarchical clustering technique is static, HCAQ-CMAC is not suitable for online training. To address this drawback, a change of clustering technique may be necessary.

As future work, the HCAQ-CMAC-based human blood glucose model would be used in the development of a blood glucose prediction system for diabetes treatment. Future applications of the HCAQ-CMAC network also includes various pattern recognition tasks [69]–[71], financial engineering [72], and biomedical domain [73]. Currently, these research endeavours are actively underway at the C2i [55]. The C2i lab undertakes intense research in the study and development of advanced brain-inspired learning memory architectures [74]–[78] for the modeling of complex, dynamic, and nonlinear systems. These techniques have been successfully applied to numerous novel applications such as automated driving [58], signature forgery detection [79], gear control for the continuous variable transmission (CVT) system in an automobile [80], fingerprint verification [81], bank failure classification and early warning system (EWS) [82], computational finance [83], [84], as well as in the biomedical engineering domain [85], [86].

APPENDIX A
HCAQ-CMAC NEURAL CORRELATES

See Table VI.

APPENDIX B
PROOF OF LEARNING CONVERGENCE

This section provides the mathematical proof of the expression $\lim_{i\to\infty}(\mathbf{G}_s)^i\mathbf{C}_a = 0$ for all $a \in \{1\cdots S\}$, given that the learning constant α satisfies the condition $0 < \alpha < 2$.

Lemma 1: Given the definition of the matrix $\mathbf{E}_s$ as

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T), \qquad s \in \{1\cdots S\}$$

and the activation mask $\mathbf{C}_s$ of the HCAQ-CMAC network for the sth input training vector as

$$\mathbf{C}_s^T = \underbrace{[c_{s,1}\ c_{s,2}\ \cdots\ c_{s,\hat{M}^J}]}_{1\times\hat{M}^J \text{ array}}\ s\in\{1\cdots S\}$$

$$c_{s,j} = \begin{cases} 1, & \text{if the } j\text{th memory cell is activated} \\ 0, & \text{otherwise.} \end{cases}$$

the $\hat{M}^J \times \hat{M}^J$ matrix $\mathbf{E}_s$ has the following properties.

Property 1: $\mathbf{E}_s$ is a symmetric matrix.

Proof:

$$\begin{aligned} \mathbf{E}_s &= \mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T \\ \mathbf{E}_s^T &= \left(\mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T\right)^T \\ &= \mathbf{I} - \alpha\left(\mathbf{C}_s\mathbf{C}_s^T\right)^T \\ &= \mathbf{I} - \alpha\left(\mathbf{C}_s^T\right)^T\mathbf{C}_s^T \\ &= \mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T \\ &= \mathbf{E}_s. \end{aligned}$$

Thus, $\mathbf{E}_s$ is a symmetric matrix. ■

Property 2: Let matrix $\mathbf{E}_s$ be denoted as $[e_{p,q}]_{\hat{M}^J\times\hat{M}^J}$. The diagonal elements of matrix $\mathbf{E}_s$ can be expressed as

$$e_{p,p} = \begin{cases} 1, & \text{if } p\text{th element of } \mathbf{C}_s \text{ is } 0 \\ 1-\alpha, & \text{if } p\text{th element of } \mathbf{C}_s \text{ is } 1 \end{cases} \qquad p\in\{1\cdots\hat{M}^J\}$$

and the nondiagonal elements of matrix $\mathbf{E}_s$ are always zero, i.e.,

$$e_{p,q} = 0 \qquad \forall p\neq q, \qquad p,q\in\{1\cdots\hat{M}^J\}.$$

Proof: According to the definition of the activation mask $\mathbf{C}_s$, as well as from the principle of the winner-take-all learning algorithm of HCAQ-CMAC, there will only be one nonzero element in $\mathbf{C}_s$ for the sth input training vector. Consequently, as the matrix $\mathbf{E}_s$ is defined as

$$\mathbf{E}_s \equiv \left(\mathbf{I} - \alpha\mathbf{C}_s\mathbf{C}_s^T\right), \qquad s\in\{1\cdots S\}$$

it follows that

$$e_{p,p} = \begin{cases} 1, & \text{if } p\text{th element of } \mathbf{C}_s \text{ is } 0 \\ 1-\alpha, & \text{if } p\text{th element of } \mathbf{C}_s \text{ is } 1 \end{cases}$$

and $e_{p,q}$ is 0 for all $p\neq q$. ■

Lemma 2: Let the matrix $\mathbf{H}$ be the multiplication result for any arbitrary $\hat{M}^J\times\hat{M}^J$ matrix $\mathbf{B}$ and the matrix $\mathbf{E}_s$ such that

$$\mathbf{H} = \mathbf{B}\mathbf{E}_s, s\in\{1\cdots S\}$$

where $\mathbf{H}$ is a $\hat{M}^J\times\hat{M}^J$ matrix. If the learning rate α satisfies the condition $0<\alpha\leq 2$, then the L_2-norm of any arbitrary rth row vector in $\mathbf{H}$ will be bounded by the L_2-norm of the corresponding rth row vector in $\mathbf{B}$. That is

$$\|\text{ row } r \text{ of } \mathbf{H}\|_2^2 \leq \|\text{ row } r \text{ of } \mathbf{B}\|_2^2 \text{ if } 0<\alpha\leq 2.$$

Proof: Let the matrix $\mathbf{H}$ be denoted as $[h_{r,q}]_{\hat{M}^J\times\hat{M}^J}$ and matrix $\mathbf{B}$ be denoted as $[b_{r,q}]_{\hat{M}^J\times\hat{M}^J}$, respectively. The L_2-norm of the rth row vector of $\mathbf{B}$ is evaluated as

$$\|\text{ row } r \text{ of } \mathbf{B}\|_2^2 = \sum_{q=1}^{\hat{M}^J}(b_{r,q})^2 = (b_{r,1})^2+(b_{r,2})^2+\cdots+(b_{r,\hat{M}^J})^2.$$

On the other hand, the L_2-norm of the rth row vector of $\mathbf{H}$ can be derived as

$$\begin{aligned} \|\text{ row } r \text{ of } \mathbf{H}\|_2^2 = \|\text{ row } r \text{ of } \mathbf{B}\mathbf{E}_s\|_2^2 &= \sum_{q=1}^{\hat{M}^J}(h_{r,q})^2 \\ &= \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,1}\right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,2}\right]^2 \\ &\quad + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,3}\right]^2 + \cdots + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,\hat{M}^J}\right]^2. \end{aligned}$$

From Property 2, it is established that all of the nondiagonal elements $e_{p,q}$ of the matrix $\mathbf{E}_s$ evaluate as zero. Furthermore, of all the diagonal elements of the matrix $\mathbf{E}_s$, exactly *one* element is equal to $(1-\alpha)$. Let this element be at the kth position in the $\mathbf{C}_s$ activation mask, i.e., $e_{k,k} = (1-\alpha)$. Substituting the value of $e_{k,k}$ into the L_2-norm of the rth row vector of $\mathbf{H}$ yields

$$\begin{aligned} &\|\text{ row } r \text{ of } \mathbf{H}\|_2^2 \\ &= \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,1}\right]^2 + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,2}\right]^2 \\ &\quad + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,3}\right]^2 + \cdots + \left[\sum_{q=1}^{\hat{M}^J} b_{r,q}e_{q,\hat{M}^J}\right]^2 \\ &= (b_{r,1})^2 + (b_{r,2})^2 + \cdots + ((1-\alpha)b_{r,k})^2 + \cdots + (b_{r,\hat{M}^J})^2 \\ &= (b_{r,1})^2 + (b_{r,2})^2 + \cdots + \underbrace{(b_{r,k})^2 + \alpha(\alpha-2)(b_{r,k})^2}_{((1-\alpha)b_{r,k})^2} \\ &\quad + \cdots + (b_{r,\hat{M}^J})^2 \\ &= \underbrace{(b_{r,1})^2 + (b_{r,2})^2 + \cdots + (b_{r,\hat{M}^J})^2}_{\|\text{row } r \text{ of } \mathbf{B}\|_2^2} + \alpha(\alpha-2)(b_{r,k})^2 \\ &= \|\text{ row } r \text{ of } \mathbf{B}\|_2^2 + \alpha(\alpha-2)(b_{r,k})^2. \end{aligned}$$

For $\|\text{ row } r \text{ of } \mathbf{H}\|_2^2 \leq \|\text{ row } r \text{ of } \mathbf{B}\|_2^2$, the term $\alpha(\alpha-2)(b_{r,k})^2$ has to be less than or equal to 0, i.e.,

$$\begin{aligned} \alpha(\alpha-2)\ \underbrace{(b_{r,k})^2}_{\text{always } +\text{ve when } b_{r,k}\neq 0} &\leq 0 \\ \alpha(\alpha-2) &\leq 0 \\ 0\leq\alpha &\leq 2. \end{aligned}$$

However, the condition $\alpha = 0$ signifies no learning and, therefore, does not apply. Hence, $\|\text{ row } r \text{ of } \mathbf{H}\|_2^2 \leq \|\text{ row } r \text{ of } \mathbf{B}\|_2^2$ when the learning constant α is $0<\alpha\leq 2$. ■

Lemma 3: Following Lemma 2, if the learning constant α satisfies the conditions $0 < \alpha < 2$ and $b_{r,k} \neq 0$, then the L_2-norm of any arbitrary rth row vector in $\mathbf{H}$ will always be smaller than the L_2-norm of the corresponding rth row vector in $\mathbf{B}$. That is

$$\| \text{ row } r \text{ of } \mathbf{H} \|_2^2 < \| \text{ row } r \text{ of } \mathbf{B} \|_2^2, \quad \text{if } 0 < \alpha < 2 \quad \text{and} \quad b_{r,k} \neq 0.$$

Furthermore, given that the learning constant α satisfies the condition $0 < \alpha < 2$, the L_2-norm of any arbitrary rth row vector in $\mathbf{H}$ will be equal to the L_2-norm of the corresponding rth row vector in $\mathbf{B}$ if and only if $b_{r,k} = 0$, where k denotes the position of the winner neuron in the activation mask $\mathbf{C}_s$, i.e.,

$$\text{if } 0 < \alpha < 2, \ \| \text{ row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{ row } r \text{ of } \mathbf{B} \|_2^2, \quad \text{iff } b_{r,k} = 0 \quad \text{and} \quad e_{k,k} = 1 - \alpha.$$

Proof: From Lemma 2, it has been established that if the learning constant α satisfies the condition $0 < \alpha \leq 2$, then $\| \text{ row } r \text{ of } \mathbf{H} \|_2^2$ will not be greater than $\| \text{ row } r \text{ of } \mathbf{B} \|_2^2$. It follows that $\alpha(\alpha - 2)(b_{r,k})^2$ of Lemma 2 is always less than zero if $0 < \alpha < 2$ and $b_{r,k} \neq 0$, where k denotes the position of the winner neuron in the activation mask $\mathbf{C}_s$. Hence

$$\| \text{ row } r \text{ of } \mathbf{H} \|_2^2 < \| \text{ row } r \text{ of } \mathbf{B} \|_2^2, \quad \text{if } 0 < \alpha < 2 \quad \text{and} \quad b_{r,k} \neq 0.$$

Consequently, the condition of

$$\| \text{ row } r \text{ of } \mathbf{H} \|_2^2 = \| \text{ row } r \text{ of } \mathbf{B} \|_2^2$$

will hold *if and only if* $b_{r,k} = 0$ for $e_{k,k} = 1 - \alpha$. ■

Lemma 4: If the learning constant α satisfies the condition $0 < \alpha < 2$, then the L_2-norm of any arbitrary rth row vector in $(\mathbf{G}_s)^i$ is bounded by the L_2-norm of the corresponding rth row vector in $(\mathbf{G}_s)^{i-1}$. That is

$$\| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 \leq \| \text{ row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2, \qquad s \in \{1 \cdots S\}.$$

Furthermore, as the training iteration i approaches infinity

$$\lim_{i \to \infty} \| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0, \qquad r \in \mathbf{K}_{\mathrm{U}}$$

where $\mathbf{U}$ denotes the entire set of the training data and $\mathbf{K}_U$ is the set of indexes of the trained cells in HCAQ-CMAC due to $\mathbf{U}$.

Proof: From the definition of the matrix $\mathbf{G}_s$ [see (34)]

$$\mathbf{G}_s = \underbrace{\mathbf{E}_{s-1}\mathbf{E}_{s-2} \ldots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \ldots \mathbf{E}_s}_{S \text{ terms}}$$

where S is the total number of input training samples and $\mathbf{G}_s$ is a $\hat{M}^J \times \hat{M}^J$ matrix. From Property 2 (Lemma 1), $\mathbf{E}_s$ is a diagonal matrix such that

$$\mathbf{E}_s = \{e_{p,p}\} = \begin{cases} 1, & \text{if } p \neq k \\ 1 - \alpha, & \text{if } p = k \end{cases} \qquad s \in \{1 \cdots S\}$$

where k is the index of the activated cell in $\mathbf{C}_s$. Hence, $\mathbf{G}_s$ is also a diagonal matrix.

From Lemma 3, if $0 < \alpha < 2$

$$\begin{aligned} &\| \text{ row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2 \\ &\quad \geq \| \text{ row } r \text{ of } (\mathbf{G}_s)^{i-1}\mathbf{E}_{s-1} \|_2^2 \\ &\quad \geq \| \text{ row } r \text{ of } ((\mathbf{G}_s)^{i-1}\mathbf{E}_{s-1})\mathbf{E}_{s-2} \|_2^2 \\ &\quad \geq \cdots \\ &\quad \geq \| \text{ row } r \text{ of } ((\mathbf{G}_s)^{i-1}\mathbf{E}_{s-1}\mathbf{E}_{s-2} \\ &\qquad \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_{s+1})\mathbf{E}_s \|_2^2 \\ &\quad \geq \| \text{ row } r \text{ of } (\mathbf{G}_s)^{i-1}\mathbf{G}_s \|_2^2 \\ &\quad \geq \| \text{ row } r \text{ of } (\mathbf{G}_s)^{i} \|_2^2 . \end{aligned}$$

Note that $\| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 1$ if $r \in \{1 \cdots \hat{M}^J\}$ is an untrained cell. On the other hand, if $r \in \mathbf{K}_{\mathrm{U}}$, it follows from Lemma 3 that $\| \text{ row } r \text{ of } (\mathbf{G}_s)^{i-1} \|_2^2 > \| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2$ as $0 < \alpha < 2$ and $g_{r,k} \neq 0$. Following from aforementioned, as the training iteration i tends to infinity

$$\lim_{i \to \infty} \| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0, \qquad r \in \mathbf{K}_{\mathrm{U}}.$$

■

Lemma 5: If the learning constant α satisfies the condition $0 < \alpha < 2$, then as the training iteration i tends to infinity and the term $(\mathbf{G}_s)^i\mathbf{C}_a$ converges to a null matrix for all $a \in \{1 \cdots S\}$. That is

$$\text{if } 0 < \alpha < 2, \qquad \lim_{i \to \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [0] \quad \forall a \in \{1 \cdots S\}, s \in \{1 \cdots S\}.$$

Proof: From Lemma 4

$$\lim_{i \to \infty} \| \text{ row } r \text{ of } (\mathbf{G}_s)^i \|_2^2 = 0, \qquad r \in \mathbf{K}_{\mathrm{U}}.$$

Hence, it follows that

$$\text{if } 0 < \alpha < 2, \qquad \lim_{i \to \infty} (\mathbf{G}_s)^i \mathbf{C}_a = [0] \qquad \forall a \in \{1 \cdots S\}.$$

■

REFERENCES

[1] F. A. Middleton and P. L. Strick, "The cerebellum: An overview," *Trends Cogn. Sci.*, vol. 27, no. 9, pp. 305–306, 1998.

[2] J. S. Albus, "Marr and Albus theories of the cerebellum: Two early models of associative memory," in *Proc. IEEE COMPCON*, 1989, pp. 577–582.

[3] J. S. Albus, "A theory of cerebellar function," *Math. Biosci.*, vol. 10, no. 1, pp. 25–61, 1971.

[4] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science*, 4th ed. New York: McGraw-Hill, 2000.

[5] D. Marr, "A theory of cerebellar cortex," *J. Physiol. London*, vol. 202, pp. 437–470, 1969.

[6] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *J. Dyn. Syst. Meas. Control*, pp. 220–227, 1975.

[7] J. S. Albus, "Data storage in cerebellar model articullation controller (CMAC)," *J. Dyn. Syst. Meas. Control*, pp. 228–233, 1975.

[8] T. Yamamoto and M. Kaneda, "Intelligent controller using CMACs with self-organized structure and its application for a process system," *IEICE Trans. Fundam. Electron. Commun. Comput.Sci.*, vol. E82-A, no. 5, pp. 856–860, 1999.

[9] S. Ku, G. A. Larsen, and S. Cetinkunt, "Fast servo control for ultra-precision machining at extremely low feed rates," *Mechatronics*, pp. 381–393, 1998.

[10] G. A. Larsen, S. Cetinkunt, and A. Donmez, "CMAC neural network control for high precision motion control in the presence of large friction," *J. Dyn. Syst. Meas. Control*, vol. 117, pp. 415–420, 1995.

[11] S. Cetinkunt and A. Donmez, "CMAC learning controller for servo control of high precision machine tools," in *Proc. Amer. Control Conf.*, San Fransico, CA, 1993, pp. 1976–1980.

[12] C. S. Lin and K. Hyongsuk, "CMAC-based adaptive critic self-learning control," *IEEE Trans. Neural Netw.*, vol. 2, no. 5, pp. 530–533, Sep. 1991.

[13] S. Commuri, S. Jagannathan, and F. L. Lewis, "CMAC neural network control of robot manipulators," *J. Robot. Syst.*, vol. 14, no. 6, pp. 465–482, 1997.

[14] H. Kano and K. Takayama, "Learning control of robotic manipulators based on neurological model CMAC," in *Proc. Triennial World Congr. Int. Federation Autom. Control*, 1990, pp. 249–254.

[15] C. M. Lin and Y. F. Peng, "Missile guidance law design using adaptive cerebellar model articulation controller," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 636–644, May 2005.

[16] A. Wahab, E. C. Tan, and H. Abut, "HCMAC amplitude spectral subtraction for noise cancellation," *Proc. Int. Conf. Neural Inf. Process.*, 2001.

[17] K. L. Huang, S. C. Hsieh, and H. C. Fu, "Cascade-CMAC neural network applications on the color scanner to printer calibration," *Proc. Int. Conf. Neural Netw.*, vol. 1, pp. 10–15, 1997.

[18] J. Ker, C. Hsu, Y. Kuo, and B. Liu, "A fuzzy CMAC model for color reproduction," *Fuzzy Sets Syst.*, vol. 91, no. 1, pp. 53–68, 1997.

[19] C. He, L. Xu, and Y. Zhang, "Learning convergence of CMAC algorithm," *Neural Process. Lett.*, vol. 14, no. 1, pp. 61–74, 2001.

[20] C. S. Lin and C. T. Chiang, "Learning convergence of CMAC technique," *IEEE Trans. Neural Netw.*, vol. 8, no. 6, pp. 1281–1292, Nov. 1997.

[21] Y. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 115–121, Jan. 1992.

[22] Y. Wei and M. Zhao, "Efective strategies for complex skill real-time learning using reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Intell. Syst. Signal Process.*, Changsa, China, 2003, pp. 388–392.

[23] X. Benavent, J. Domingo, F. Vegara, and J. Pelechano, "Two suggestions for efficient implementation of CMAC's," in *Proc. IEEE Int. Symp. Intell. Control*, Rio, Patras, Greece, 2000, pp. 309–314.

[24] A. Menozzi and M. Chow, "On the training of a multi-resolution CMAC neural network," in *Proc. Int. Conf. Ind. Electron. Control Instrum.*, 1997, vol. 3, pp. 1130–1135.

[25] J. Moody, "Fast-learning in multi-resolution hierarchies," in *Adv. Neural Infor. Processing Syst.*. San Mateo, CA: Morgan Kauffman, 1989, vol. 14, pt. 1, pp. 29–38.

[26] M. F. Yeh and H. C. Lu, "On-line adaptive quantization input space in CMAC neural network," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2002, vol. 4, p. 6.

[27] X. Gao, C. Wang, X. M. Gao, and S. J. Ovaska, "A new CMAC neural network model with adaptive quantization input layer," in *Proc. Int. Conf. Signal Process.*, 1996, vol. 2, pp. 1417–1420.

[28] H. Kim and C. S. Lin, "Use of adaptive resolution for better CMAC learning," in *Proc. Int. Joint Conf. Neural Netw.*, 1992, vol. 117, pp. 517–522.

[29] H. M. Lee, C. M. Chen, and Y. F. Lu, "A self-organizing hcmac neural-netwoek classifier," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 15–27, Jan. 2003.

[30] J. Hu and F. Pratt, "Self-organizing CMAC neural networks and adaptive dynamic control," in *Proc. IEEE Int. Symp. Intell. Control/Intell. Syst. Semiotics*, 1999, pp. 259–265.

[31] P. Shaw, D. Greenstein, J. Lerch, L. Clasen, R. Lenroot, N. Gogtay, A. Evans, J. Rapoport, and J. Giedd, "Intellectual ability and cortical development in children and adolescents," *Nature*, vol. 440, no. 3, pp. 676–679, 2006.

[32] W. T. Thach, "What is the role of the cerebellum in motor learning and cognition?," *Trends Cogn. Sci.*, vol. 27, no. 9, pp. 331–337, 1998.

[33] F. A. Middleton and P. L. Strick, "Cerebellar output: Motor and cognitive channels," *Trends Cogn. Sci.*, vol. 27, no. 9, pp. 348–354, 1998.

[34] T. Tyrrell and D. Willshaw, "Cerebellar cortex: Its simulation and the relevance of Marr's theory," *Philosoph. Trans.: Biol. Sci.*, vol. 336, no. 1277, pp. 239–257, 1992.

[35] T. Jessell, , E. R. Kandel, J. H. Schwartz, and T. M. Jessell, Eds., "Development of the nervous system," in *Essentials of Neural Science and Behavior*. New York: McGraw-Hill, 1996, pp. 89–110.

[36] J. G. Nicholls, A. R. Martin, B. G. Wallace, and P. A. Fuchs, *From Neuron to Brain*, 4th ed. Sunderland, MA: Sinauer Associates Inc., 2001.

[37] J. H. Kaas, R. J. Nelson, M. Sur, and C. S. Lin, "Multiple representations of the body within the primary somatosensory cortex of primates," *Science*, vol. 204, pp. 521–523, 1979.

[38] J. H. Kaas, R. J. Nelson, M. Sur, and M. M. MerzenichF. O. Schmitt, F. G. Worden, G. Adelman, and S. G. Dennis, "Organization of somatosensory cortex in primates," in *The Organization of the Cerebral Cortex: Proceedings of a Neuroscience Research Program Colloquium*. Cambridge, MA: MIT Press, 1981, pp. 237–261.

[39] D. H. Hubel and T. N. Weisel, "Receptive fields, binocular interaction and functional architecture int he cat's visual cortex," *J. Physiol. London*, vol. 160, pp. 106–154, 1962.

[40] M. Ito, "Mechanisms of motor learning in the cerebellum," *Brain Res.*, vol. 886, pp. 237–245, 2000.

[41] E. D. Schutter, "A new functional tole for cerebellar long term depression," *Progr. Brain Res.*, vol. 114, pp. 529–542, 1997.

[42] J. C. Houk, J. T. Buckingham, and A. G. Barto, "Models of the cerebellum and motor learning," *Behav. Brain Sci.*, vol. 19, no. 3, pp. 368–383, 1996.

[43] J. A. Kleim, E. Lussnig, E. R. Schwars, T. A. Comery, and W. T. Greenough, "Synaptogenesis and FOS expression in the motor cortex of the adult rat after motor skill learning," *J. Neurosci.*, vol. 16, no. 14, pp. 4529–4535, 1996.

[44] J. A. Kleim, R. A. Swain, K. A. Amstrong, R. M. A. Napper, T. A. Jones, and W. T. Greenough, "Selective synaptic plasticity within the cerebellar cortex following complex motor skill learning," *Neurobiol. Learn. Memory*, vol. 69, pp. 274–289, 1998.

[45] J. A. Kleim, M. A. Pipitone, C. Czerlanis, and W. T. Greenough, "Structural stability within the lateral cerebellar nucleus of the rat following complex motor learning," *Neurobiol. Learn. Memory*, vol. 69, pp. 290–306, 1998.

[46] K. D. Federmeier, J. A. Kleim, and W. T. Greenough, "Learning-induces multiple synapse formation in rat cerebellar cortex," *Neurosci. Lett.*, vol. 332, pp. 180–184, 2002.

[47] J. S. Albus, *Brains, Behavior and Robotics*. New York: BYTE Books/ McGraw-Hill, 1981.

[48] S. H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher-order CMAC neural networks," *IEEE Control Syst. Mag.*, vol. 12, no. 2, pp. 23–30, Apr. 1992.

[49] G. Horvath, "CMAC: Reconsidering an old neural network," in *Proc. Intell. Control Syst. Signal Process.* , 2003, pp. 173–178.

[50] G. Horvath, "CMAC neural network as an SVM with B-Spline kernel functions," in *Proc. Instrum. Meas. Technol. Conf.*, 2003, pp. 1108–1113.

[51] J. H. Nie and D. A. Linkens, "A fuzzified CMAC self-learning controller," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1993, vol. 1, pp. 500–505.

[52] J. Sim, W. L. Tung, and C. Quek, "FCMAC-Yager: A novel Yager inference scheme based fuzzy CMAC," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1394–1410, Nov. 2006.

[53] J. H. Ward, "Hierarchical grouping to optimize an objective function," *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 236–244, 1963.

[54] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[55] Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore [Online]. Available: http://www.c2i.ntu.edu.sg

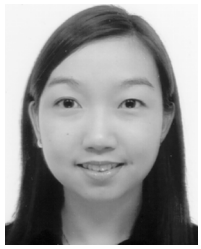
[56] H. Eichenbaum, *The Cognitive Neuroscience of Memory: An Introduction*. Oxford, U.K.: Oxford Univ. Press, 2002.

[57] M. C. Nechyba and Y. Xu, "Human control strategy: Abstraction, verification, and replication," *IEEE Control Syst. Mag.*, vol. 17, no. 5, pp. 48–61, Oct. 1997.

[58] M. Pasquier, C. Quek, and M. Toh, "Fuzzylot: A self-organizing fuzzy neural rule-based pilot system for automated vehicle," *Neural Netw.*, vol. 14, no. 8, pp. 1099–1112, 2001.

[59] L. Fletcher, G. Hraban, P. Huang, B. Srinivasan, and R. Venook, "Feasibility of an implanted, closed-loop, blood-glucose control device," *Immunology*, vol. 230, 2001.

[60] L. M. Schetky, P. Jardine, and F. Moussy, "A closed loop implantable artificial pancreas using thin film nitinol MEMS pumps," in *Proc. Int. Conf. Shape Memory Superelastic Technol.*, 2003, pp. 555–561.
[61] J. T. Sorensen, "A physiologic model of glucose metabolism in man and its use to design and assess improved insulin therapies for diabetes," Ph.D. dissertation, Dept. Chem. Eng., MIT, Cambridge, MA, 1985.
[62] M. W. Schwartz and D. Porte, Jr., "Diabetes, obesity and the brain," *Science*, vol. 307, pp. 375–379, 2005.
[63] D. Porte, Jr., D. G. Baskin, and M. W. Schwartz, "Insulin signaling in the central nervous system," *Diabetes*, vol. 54, no. 5, pp. 1264–1276, 2005.
[64] A. Pocai, T. K. T. Lam, R. Gutierrez-Juarez, S. Obici, G. J. Schwartz, J. Bryan, L. Aguilar-Bryan, and L. Rossetti, "Hypothalamic K_{ATP} channels control hepatic glucose production," *Nature*, vol. 434, pp. 1026–1031, 2005.
[65] S. Obici, B. B. Zhang, G. Karkanias, and L. Rossetti, "Hypothalamic insulin signaling is required for inhibition of glucose production," *Nature Med.*, vol. 8, no. 12, pp. 1376–1382, 2002.
[66] S. J. Fisher and C. R. Kahn, "Insulin signaling is required for insulin's direct and indirect action on hepatic glucose production," *J. Clinical Investigat.*, vol. 111, no. 4, pp. 463–468, 2003.
[67] Illinois Inst. Technol., "Glucosim: A web-based educational simulation package for glucose-insulin levels in the human body," Chicago, IL [Online]. Available: http://216.47.139.198/glucosim/gsimul.html
[68] Health Promotion Board Singapore [Online]. Available: http://www.hpb.gov.sg
[69] Y. H. Liu and Y. T. Chen, "Face recognition using total margin-based adaptive fuzzy support vector machines," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 178–192, Jan. 2007.
[70] E. M. Kussul, T. N. Baidyk, D. C. Wunsch II, O. Makeyev, and A. Martn, "Permutation coding technique for image recognition systems," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1566–1579, Nov. 2006.
[71] G. Pajares, "A Hopfield neural network for image change detection," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1250–1264, Sep. 2006.
[72] R. Gencay and R. Gibson, "Model risk for European-style stock index options," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 193–202, Jan. 2007.
[73] Y. Gao and M. J. Er, "An intelligent adaptive control scheme for post-surgical blood pressure regulation," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 475–483, Mar. 2005.
[74] C. Quek and R. W. Zhou, "POPFNN: A pseudo outer-product based fuzzy neural network," *Neural Netw.*, vol. 9, no. 9, pp. 1569–1581, 1996.
[75] W. L. Tung and C. Quek, "GenSoFNN: A generic self-organizing fuzzy neural network," *IEEE Trans. Neural Netw.*, vol. 13, no. 5, pp. 1075–1086, Sep. 2002.
[76] K. Ang, C. Quek, and M. Pasquier, "POPFNN-CRI(S): Pseudo outer product based fuzzy neural network using the compositional rule of inference and singleton fuzzifier," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 6, pp. 838–849, Dec. 2003.
[77] W. L. Tung and C. Quek, "Falcon: Neuro fuzzy control and decision systems using FKP and PFKP clustering algorithms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 1, pp. 686–695, Feb. 2004.
[78] K. K. Ang and C. Quek, "RSPOP: Rough set-based pseudo outer-product fuzzy rule identification algorithm," *Neural Comput.*, vol. 17, no. 1, pp. 205–243, 2005.
[79] R. W. Zhou and C. Quek, "Antiforgery: A novel pseudo-outer product based fuzzy neural network driven signature verification system," *Pattern Recognit. Lett.*, vol. 230, no. 14, pp. 1795–1816, 2002.
[80] K. K. Ang, C. Quek, and A. Wahab, "MCMAC-CVT: A novel on-line associative memory based CVT transmission control system," *Neural Netw.*, vol. 15, pp. 219–236, 2001.
[81] C. Quek, B. Tan, and V. Sagar, "POPFNN-based fingerprint verification system," *Neural Netw.*, vol. 14, pp. 305–323, 2001.
[82] W. L. Tung, C. Quek, and P. Y. K. Cheng, "GenSo-EWS: A novel neural-fuzzy based early warning system for predicting bank failures," *Neural Netw.*, vol. 17, no. 4, pp. 567–587, 2004.
[83] K. Quah and C. Quek, "Maximum reward reinforcement learning: A non-cumulative reward criterion," *Expert Syst. Appl.*, vol. 31, no. 2, pp. 351–359, 2006.
[84] K. K. Ang and C. Quek, "Stock trading using RSPOP: A novel rough set based neuro-fuzzy approach," *IEEE Trans. Neural Netw.*, vol. 17, no. 5, pp. 1301–1315, Sep. 2006.
[85] T. Z. Tan, G. S. Ng, and C. Quek, "Ovarian cancer diagnosis by hippocampus and neocortex-inspired learning memory structures," *Neural Netw.*, vol. 18, no. 5–6, pp. 818–825, 2005.
[86] W. L. Tung and C. Quek, "GenSo-FDSS: A neural-fuzzy decision support system for pediatric all cancer subtype identification using gene expression data," *Artif. Intell. Medicine*, vol. 33, no. 1, pp. 61–88, 2005.

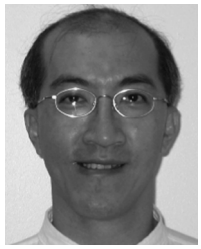


S. D. Teddy received the B.Eng. degree (first-class honors) in computer engineering from Nanyang Technological University (NTU), Singapore, in 2003, where she is currently working towards the Ph.D. degree at the Centre for Computational Intelligence, School of Computer Engineering.

Her current research interests include the cerebellum and its computational model, artificial neural networks, the study of brain-inspired learning memory systems, and autonomous control of biophysiological processes.

E. M.-K. Lai (M'82–SM'95) received the B.E. (honors) and Ph.D. degrees from the University of Western Australia, Perth, Australia, both in electrical engineering, in 1982 and 1991, respectively.

Currently, he is a Faculty Member of the Institute of Information Sciences and Technology, Massey University, Wellington, New Zealand. Previously, he has been a Faculty Member of the Department of Electrical and Electronic Engineering, The University of Western Australia, from 1985 to 1990, the Department of Information Engineering, the Chinese University of Hong Kong, from 1990 to 1995, Edith Cowan University, Perth, Australia, from 1995 to 1998, and the School of Computer Engineering, Nanyang Technological University, Singapore, from 1999 to 2006. His current research interests include artificial neural networks, digital signal processing, and information theory.



C. Quek received the B.Sc. degree in electrical and electronics engineering and the Ph.D. degree in intelligent control from Heriot-Watt University, Edinburgh, Scotland, U.K., in 1986 and 1990, respectively.

Currently, he is an Associate Professor and a member of the Centre for Computational Intelligence, formerly the Intelligent Systems Laboratory, School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include intelligent control, neurocognitive architectures, artificial intelligence in education, neural networks, fuzzy systems, fuzzy rule-based systems, and genetic algorithms and brain-inspired neurocognitive applications in computational finance and biomedical engineering.

Hierarchically Clustered Adaptive Quantization CMAC and Its Learning Convergence

Lai, Edmund M-K.

2007-11-01